



Application Server Help

SAP BusinessObjects Strategy Management 10.0

Target Audience

- Technical Consultants
- System Administrators

PUBLIC

Document version: 2.3 – 11/7/2011



SAP AG
Neurottstraße 16
69190 Walldorf
Germany
T +49/18 05/34 34 24
F +49/18 05/34 34 20
www.sap.com

© Copyright 2011 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Excel, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
IBM, DB2, DB2 Universal Database, System i, System i5, System p, System p5, System x, System z, System z10, System z9, z10, z9, iSeries, pSeries, xSeries, zSeries, eServer, z/VM, z/OS, i5/OS, S/390, OS/390, OS/400, AS/400, S/390 Parallel Enterprise Server, PowerVM, Power Architecture, POWER6+, POWER6, POWER5+, POWER5, POWER, OpenPower, PowerPC, BatchPipes, BladeCenter, System Storage, GPFS, HACMP, RETAIN, DB2 Connect, RACF, Redbooks, OS/2, Parallel Sysplex, MVS/ESA, AIX, Intelligent Miner, WebSphere, Netfinity, Tivoli and Informix are trademarks or registered trademarks of IBM Corporation.
Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.
Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.
Oracle is a registered trademark of Oracle Corporation.
UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.
HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.
Java is a registered trademark of Sun Microsystems, Inc.
JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.
SAP, R/3, SAP NetWeaver, Duet, PartnerEdge, ByDesign, SAP BusinessObjects Explorer, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.
Business Objects and the Business Objects logo, BusinessObjects, Crystal Reports, Crystal Decisions, Web Intelligence, Xcelsius, and

other Business Objects products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Business Objects Software Ltd. in the United States and in other countries.

Sybase and Adaptive Server, iAnywhere, Sybase 365, SQL Anywhere, and other Sybase products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of Sybase, Inc. Sybase is an SAP company.

All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

Disclaimer

Some components of this product are based on Java™. Any code change in these components may cause unpredictable and severe malfunctions and is therefore expressly prohibited, as is any decompilation of these components.

Any Java™ Source Code delivered with this product is only to be used by SAP's Support Services and may not be modified or altered in any way.

Legal Software Terms

Terms for Included Open Source Software

This SAP software contains also the third party open source software products listed below. Please note that for these third party products the following special terms and conditions shall apply.

Documentation in the SAP Service Marketplace






You can find this document at the following address:

<http://service.sap.com/instguides>

Typographic Conventions

Type Style	Represents
Example Text	Words or characters that appear on the screen. These include field names, screen titles, pushbuttons as well as menu names, paths and options. Cross-references to other documentation
Example text	Emphasized words or phrases in body text, titles of graphics and tables
EXAMPLE TEXT	Names of elements in the system. These include report names, program names, transaction codes, table names, and individual key words of a programming language, when surrounded by body text, for example, SELECT and INCLUDE.
Example text	Screen output. This includes file and directory names and their paths, messages, names of variables and parameters, source code as well as names of installation, upgrade and database tools.
Example text	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
<Example text>	Variable user entry. Pointed brackets indicate that you replace these words and characters with appropriate entries.
EXAMPLE TEXT	Keys on the keyboard, for example, function keys (such as F2) or the ENTER key.

Icons

Icon	Meaning
	Caution
	Example
	Note
	Recommendation
	Syntax

Document History

The following table provides an overview of the most important document changes.

Version	Date	Description
1.0	June 16, 2011	First version
2.0	August 11, 2011	Changed default value of MAXSETS in lsserver.ini from 1,000 to 5,000
2.1	September 13, 2011	Made corrections to KILL, WRITE. Added CLEAN WORK DATABASES (Supervisor) command.
2.2	October 13, 2011	Added EXHIBIT MEASURE NONVIRTUAL, CREATE variable [MIN MAX], SET VARIABLE [MIN MAX]
2.3	November 7, 2011	Corrected the MAXLOGIN default on CHANGEUSER and CREATE USER.

Contents

1	Introduction	15
2	Getting Started	16
2.1	Starting Application Server Before Creating Users or Databases	16
2.2	Starting Application Server with a User and a Database ..	16
2.3	Exiting from Application Server from the Task Bar	16
2.4	Exiting from Application Server from the IDQL Tab	17
2.5	Creating a Remote Server Connection	17
2.6	Running in Batch from the Command Line	18
2.7	Running Application Server Jobs in Batch from a UNIX Server	19
3	Application Server User Interface	20
3.1	Application Server Administrator Windows	20
3.2	Executing an IDQL Command	21
3.3	Syntax Conventions	22
3.4	Controlling Your View	24
3.5	Working with Items	30
3.6	Working with Data in the Data View Tab	37
3.7	Working with Link IDs	44
3.8	Users, Security and Database Access	49
4	Application Server at the Command Level	54
4.1	Executing Commands in Application Server	54
4.2	Running Application Server While Executing DOS Batch Jobs	54
4.3	Building an Application Server Database	54
4.4	Databases and Users	55
4.5	Dimensions	59
4.6	Attributes	64
4.7	Variables and Reading in Data	89
4.8	Data Calculation, Consolidation and Rollup	95
4.9	Date Ranges and Fiscal Year Settings	100
4.10	Periodicities	101
4.11	Time Templates	108
4.12	User-Defined Hierarchies	124
4.13	Security and Access	133
4.14	Procedure editor	137

4.15	Importing data	138
4.16	Frequently Asked Questions	139
5	Administration and Configuration	143
5.1	Debugging	143
5.2	Maximums and Limits.....	146
5.3	Creating a Dump File with Multiple Partitions.....	150
5.4	Configuring Application Server for International Use.....	151
6	Application Server Command Reference.....	152
6.1	ABS()	152
6.2	ACCESS DBASE.....	152
6.3	ACCESS EXTERNAL.....	153
6.4	ACCESS LSLINK	155
6.5	ACCUM()	156
6.6	ACROSS/DOWN	157
6.7	ACRSLIH().....	159
6.8	ACRSPP().....	160
6.9	ACRSRP().....	160
6.10	ADD ACCESS (Supervisor)	160
6.11	ADD DATABASE (Supervisor)	161
6.12	ADD USER (Supervisor)	162
6.13	ALL-ENDALL	162
6.14	ALLOCATE (Dimension or CONSTRUCT).....	163
6.15	ANALYZE	163
6.16	ANALYZE: Analytics Options	164
6.17	ANALYZE: Forecasting Options	166
6.18	ANALYZE: Regression Option	167
6.19	ANALYZE: Smoothing Options.....	170
6.20	ANALYZE: Statistical Options	171
6.21	ASK.....	172
6.22	ATTACH	177
6.23	ATTRIBUTE	177
6.24	BEGIN-END	180
6.25	BYPERIOD-ENDBYPERIOD.....	180
6.26	CALCULATE	180
6.27	CHANGE DATABASE (Supervisor)	183
6.28	CHANGE USER (Supervisor)	184
6.29	CHECKPOINT	186
6.30	CLASS (Dimension or CONSTRUCT).....	187

6.31	CHG().....	188
6.32	CLEAN WORK DATABASES (Supervisor).....	188
6.33	CLEAR.....	189
6.34	CLEAR STATUS	190
6.35	COLHEADING-ENDCOLHEADING (Report).....	190
6.36	COMPILE.....	191
6.37	COMPILE ATTRIBUTE	192
6.38	CONSOLIDATE	193
6.39	CONSTRUCT (Access).....	195
6.40	CONSTRUCT ATTRIBUTE	198
6.41	COPY	203
6.42	CREATE Variable	204
6.43	CREATE (Access) or LSS CREATE	209
6.44	CREATE DATABASE (Supervisor)	210
6.45	CREATE Dimension User_Defined_Hierarchy	213
6.46	CREATE GROUP (Supervisor)	215
6.47	CREATE USER (Supervisor)	215
6.48	CREATE VERSION	216
6.49	CSWITCH	216
6.50	DECONSTRUCT.....	218
6.51	DEFINE SYNONYM (Access).....	218
6.52	DESCRIPTION (Access External)	219
6.53	Description	219
6.54	DETACH	220
6.55	DIFF().....	222
6.56	Using DIFF and ACCUM	222
6.57	DIMENSION.....	223
6.58	DIRECTORY.....	225
6.59	DISABLE (Supervisor)	226
6.60	DISPLAY.....	227
6.61	DISPLAY (Logic).....	227
6.62	DOCUMENT	228
6.63	DO-ENDDO.....	229
6.64	DUMP.....	229
6.65	DUMP (Supervisor)	232
6.66	ECHO	233
6.67	ENABLE (Supervisor)	233
6.68	END.....	234
6.69	EXECUTE	234

6.70	EXHIBIT	235
6.71	EXIT	289
6.72	EXP()	290
6.73	EXTEND	290
6.74	EXTEND (Supervisor)	292
6.75	FOOTNOTE-ENDFOOTNOTE (Report)	292
6.76	FOR	292
6.77	FORCE (Supervisor)	293
6.78	FREE (Supervisor)	294
6.79	GET ENVIRONMENT	295
6.80	HEADING-ENDHEADING (Report)	296
6.81	HIERARCHY (Dimension)	297
6.82	IF-OTHERWISE	299
6.83	IN (OF) FROM VERSION	299
6.84	INDEX-CASE-ENDINDEX	300
6.85	INDEX USER-CASE-ENDINDEX	301
6.86	INPUT (Dimension)	303
6.87	INT()	304
6.88	IRR()	305
6.89	IS	305
6.90	JOB	306
6.91	KEY (Dimension)	307
6.92	KILL (Supervisor)	307
6.93	LAG()	308
6.94	LAST()	309
6.95	LATEST	309
6.96	LEAD()	309
6.97	LEVEL (Dimension)	310
6.98	LIST	311
6.99	LN()	312
6.100	LOAD	312
6.101	LOAD (Supervisor)	313
6.102	LOG()	313
6.103	LOGIC	314
6.104	LOOP-ENDLOOP	314
6.105	MASK-ENDMASK	315
6.106	MAX()	315
6.107	MEAN()	316
6.108	MEAN2()	316

6.109	MIN()	317
6.110	MORTGAGE()	317
6.111	MOVING()	318
6.112	MOVING2()	318
6.113	NCASES()	319
6.114	NDAYS()	320
6.115	NEWPAGE	320
6.116	NOCALC-ENDNOCALC	320
6.117	NPV()	321
6.118	NROWS()	321
6.119	OBS()	321
6.120	ON	322
6.121	ORDER	322
6.122	ORDER (Report)	326
6.123	OUTPUT	327
6.124	OUTPUT (Dimension)	328
6.125	PARHEADING-ENDPARHEADING (Report)	329
6.126	PEEK (Access)	329
6.127	PERCHG()	330
6.128	PERIODSAVAILABLE()	331
6.129	PREFACE (Report)	331
6.130	PRINT	332
6.131	PROCEDURE	333
6.132	PURGE	333
6.133	QUIT	334
6.134	RANDOM()	335
6.135	RANK()	335
6.136	READ (Access)	336
6.137	RECONSTRUCT (Access)	339
6.138	RECOVER	339
6.139	REMOVE	340
6.140	REMOVE ACCESS (Supervisor)	343
6.141	REMOVE DATABASE (Supervisor)	344
6.142	REMOVE GROUP (Supervisor)	344
6.143	REMOVE PROTECTION (Supervisor)	344
6.144	REMOVE USER (Supervisor)	345
6.145	RENAME	345
6.146	REPEAT-UNTIL	346
6.147	REPORT	346

6.148	RESTORE CUSTOM	350
6.149	RESULT (Dimension)	350
6.150	RETURN	351
6.151	ROLLUP	352
6.152	ROUND()	358
6.153	ROWS(Report)	358
6.154	SAVE (Access External)	360
6.155	SAVE CONTROL	360
6.156	SAVE CUSTOM	360
6.157	SAVE DATA	362
6.158	SAVE STATUS	363
6.159	SD()	364
6.160	SELECT <dimension>	364
6.161	SELECT <attribute>	367
6.162	SELECT measures	370
6.163	SELECT <variables>	371
6.164	SELECT VERSION	375
6.165	SET	375
6.166	SET BUFFERS	376
6.167	SET CELLS	377
6.168	SET CHARSET	377
6.169	SET CONTROL	378
6.170	SET DATE	379
6.171	SET DEFAULT	379
6.172	SET DEFAULT Periodicity	381
6.173	SET DENSE	383
6.174	SET DETAIL	383
6.175	SET-ENDSET (Report)	385
6.176	SET ENVIRONMENT	391
6.177	SET FISCAL	392
6.178	SET IGNOREMISSING MEMBERS	395
6.179	SET INDENTATION	395
6.180	SET LABEL	396
6.181	SET LATEST EARLIEST	396
6.182	SET MEMORY	397
6.183	SET MESSAGE	398
6.184	SET PERIOD	398
6.185	SET ONERROR	399
6.186	SET periodicity	400

6.187	SET PRINTER	400
6.188	SET PROGRESS	400
6.189	SET REASSURANCE	401
6.190	SET REREAD	401
6.191	SET QUERY VARIABLE	401
6.192	SET SELECT	402
6.193	SET SHARE INTERVAL.....	402
6.194	SET SHORT LONG	402
6.195	SET SPARSE	403
6.196	SET VARIABLES	404
6.197	SHOW	408
6.198	SHOW (Access External).....	413
6.199	SHOW (Supervisor).....	413
6.200	SKIP (Report).....	417
6.201	SNAPSHOT	417
6.202	SOLVE-ENDSOLVE	421
6.203	SQRT().....	421
6.204	STATUS.....	421
6.205	SUBTOTAL-ENDSUBTOTAL (Report)	422
6.206	SUBTOTAL INCLUDE (Report).....	423
6.207	SUBTRACT DATABASE (Supervisor)	423
6.208	SUM (Dimension)	424
6.209	SUMMARY-ENDSUMMARY (Report)	424
6.210	SUPERVISOR.....	425
6.211	SWITCH	425
6.212	SYNC	426
6.213	SYNONYM	427
6.214	TABLE().....	428
6.215	TABS (Report)	428
6.216	TABS SINGLE (Report).....	429
6.217	TEXT (Report)	429
6.218	TIME.....	430
6.219	TITLE (Report)	431
6.220	TOTAL().....	432
6.221	TRACE	432
6.222	TYPE	434
6.223	UDASH (Report)	434
6.224	UDATA (Report).....	435
6.225	UNAME (Report)	435

6.226	USE	435
6.227	USE (Access)	436
6.228	UTEXT (Report)	437
6.229	VALUE()	437
6.230	VERSION	438
6.231	WHEN-ENDWHEN	438
6.232	WHILE-ENDWHILE	439
6.233	WRITE (Access)	440
6.234	XRAY (Supervisor)	440
6.235	YIELD()	441
7	Working with Hybrid OLAP	442
7.1	What Is a Schema?	442
7.2	How a Schema is Structured	442
7.3	Creating a Schema	443
7.4	Creating a Hybrid OLAP Model	443
7.5	Using Fact Tables in External Applications	457
7.6	Working with SCHEMA Subsystem	457
7.7	Importing	458
7.8	Deleting a Schema	459
7.9	Viewing Definitions	459
7.10	Working with Dimensions, Variables and Data	459
8	Schema Subsystem Command Reference	461
8.1	CACHE (Schema)	461
8.2	CONSOLIDATE {PENDING INCREMENTAL} (Schema)	462
8.3	DELTA (Schema)	464
8.4	DROP BASE (Schema)	464
8.5	DROP DIMENSION (Schema)	465
8.6	DROP REFERENCES (Schema)	465
8.7	DROP SCHEMA (Schema)	466
8.8	DROP VARIABLE (Schema)	466
8.9	EXPORT BASE (Schema)	466
8.10	EXPORT DATA (Schema)	469
8.11	EXPORT DIMENSION (Schema)	471
8.12	EXPORT MASTER (Schema)	473
8.13	EXPORT SCHEMA (Schema)	474
8.14	EXPORT SKELETON (Schema)	477
8.15	EXPORT VARIABLE (Schema)	478
8.16	IMPORT DATA (Schema)	479

8.17	IMPORT DIMENSION (Schema).....	480
8.18	IMPORT DIMENSION (SAP NetWeaver BI Connector)...	480
8.19	IMPORT SCHEMA (Schema)	482
8.20	IMPORT SCHEMA (SAP NetWeaver BI Connector)	482
8.21	IMPORT TIME (SAP NetWeaver BI Connector).....	484
8.22	IMPORT VARIABLES (Schema)	485
8.23	IMPORT VARIABLES (SAP NetWeaver BI Connector) ..	485
8.24	IMPORT QUERY VARIABLES (SAP NetWeaver BI Connector)	486
8.25	PREFIX (Schema)	486
8.26	SCHEMA (Schema)	487
8.27	SET LABEL (Schema)	487
8.28	SET VARIABLE (Schema).....	488
8.29	SPY (Schema).....	488
8.30	TRACKER INSERT (Schema)	489
8.31	TRIGGER (Schema).....	489
8.32	VIEW CACHE (Schema)	490
8.33	VIEW CONNECTION (Schema).....	490
8.34	VIEW DIMENSION (Schema)	490
8.35	VIEW PREFIX (Schema).....	491
8.36	VIEW <rowsettype> ROWSET	491
8.37	VIEW SPANS TYPE	492
8.38	VIEW TIME RANGE	492
8.39	VIEW SCHEMA (Schema)	492
8.40	VIEW SPY (Schema).....	493
8.41	VIEW VARIABLE (Schema)	493
8.42	DROP DATA (Schema).....	493
8.43	CONNECT (Schema)	493
8.44	IMPORT FISCAL (Schema)	494
8.45	SQL (Schema).....	494
8.46	ORDER DRILLTHRU (Schema)	494
9	Schema Table Reference	496
9.1	Schema Map	496
9.2	Consolidations Table	497
9.3	Fact Tables	497
9.4	Fact Delta Tables.....	498
9.5	Dimension Types Table	498
9.6	Dimensions Table	499

9.7	Dimension Tables	500
9.8	Dimension Delta Tables.....	501
9.9	By Dimensions Table.....	501
9.10	Periodicities Table	502
9.11	Master Table	502
9.12	Month Names Table	503
9.13	Day Names Table	503
9.14	Format Table.....	503
9.15	Period Format Table	504
9.16	Variables Table.....	504
9.17	Dimension Levels Table	506
9.18	Tracker Table.....	506
9.19	Reference Tables	507
9.20	Attribute Tables.....	508
9.21	Table Parameters Table Reference	508
10	Hybrid OLAP Reference Topics	519
10.1	Extensions and Constraints to Application Server	519
10.2	Maintaining Your Environment	520
11	Transformer	526
11.1	What Is the Transformer?.....	526
11.2	Steps to Take Before Running the Transformer	526
11.3	Creating a Transformer Parameter File.....	526
11.4	Running the Transformer	527
11.5	Parameter File Reference	527

1 Introduction

About Application Server

Application Server provides analytical modeling capabilities through time-based analysis and dynamic dimensions, cross tabs, and user-defined Hierarchies. It includes a library of built-in functions that support forecasting techniques, correlation methods, and regression analyses.

Use Application Server Administrator to build, maintain, and administer your Application Server models using DQL commands.

In addition, you can use Hybrid OLAP with Application Server to integrate relational database technology with Application Server's multidimensional database. Hybrid OLAP fully integrates relational database technology with Application Server's multidimensional database to create an optimal data warehouse/data mart environment. It greatly increases the scalability of Application Server by extending support of large analysis models. Hybrid OLAP also supports large numbers of dimension members. You can perform complex analyses on large data sets and work with analysis models previously beyond the scope of Application Server.

You access Hybrid OLAP through the SCHEMA subsystem, which provides certain Application Server commands that allow you to communicate with the relational database schema from the IDQL command line.

Hybrid OLAP has flexible data storage, allowing you to store your data in the following forms:

Base data in the RDBMS

Preconsolidated data in the RDBMS

Preconsolidated data in the RDBMS, replicated to Application Server

Consolidated on the fly

Attribute data in the RDBMS

Input data at output levels in the RDBMS

Preconsolidated data in Application Server

You can combine different forms of data storage to get the storage solution that best suits your system, and change these forms as your model evolves.

When you create a Hybrid OLAP schema from an Application Server multidimensional model, the variables in the model are assigned the Drillthru attribute. This means that the data for the variable can be stored anywhere in the Hybrid OLAP analysis environment, whether this is in Application Server itself, or in relational tables. The actual location of the data is transparent to the end user. This means you can store data wherever it makes the most sense. Typically, input level data is stored in relational tables, while high-level consolidations are stored in Application Server.

Storage independent analysis also means you can handle databases normally beyond the scope of Application Server.

Target Groups

Technical Consultants

System Administrators

Solution Consultants

Business Process Owner

Support Specialist

2 Getting Started

2.1 Starting Application Server Before Creating Users or Databases

Procedure

1. On the Windows taskbar, choose *Start -> All Programs -> SAP BusinessObjects -> Strategy Management*.
2. When the Application Server login dialog box appears, enter the user names that already exist in MASTERDB. Check your release notes for more information about these special function logins:
 - admin
 - super
 - supervisor
3. Click OK.

2.2 Starting Application Server with a User and a Database

Procedure

1. On the Windows taskbar, choose *Start -> All Programs -> SAP BusinessObjects -> Strategy Management*.

If you have not previously specified login parameters in the Options dialog box, the Application Server login dialog box is displayed by default when the Application Server program starts.
2. When the Application Server login dialog box appears, do the following:
 - a. In the User text box, type the user name.
 - b. In the Password text box, type the user's password.
 - c. In the Server text box, enter the name of the default database for the specified user.
3. Click OK.

Note: Once you are logged in to Application Server, the current model name and access mode are shown on the status bar. Learn how to change the dimensional model and access mode.

Application Server is installed with language files based on the language set for your system. For information about overriding regional settings for currency and dates and other things, see *Configuring Application Server for international use*.

2.3 Exiting from Application Server from the Task Bar

Procedure

1. From the File menu, choose Exit Application Server. Or, double-click the icon at the top-left corner of the application window.

2. (Optional) If you have made changes that you have not yet saved, Application Server prompts you to save the changes. Do one of the following:
 - Click Yes to save the changes and exit.
 - Click No to exit without saving the changes.
 - Click Cancel to cancel the exit request.
3. (Optional) If "Confirm before exiting Application Server" is selected in the Options dialog box, Application Server prompts you to save your Work database. Do one of the following:
 - Click Yes to save your Work database and exit.
 - Click No to exit without saving the Work database.

2.4 Exiting from Application Server from the IDQL Tab

Procedure

Enter the EXIT or EXIT CLEAR command in the Input window.

Note: By default when you close Application Server by choosing Exit from the File menu, an EXIT command is executed. However, by adding the line EXITCLEAR=Yes to the Windows section of LSSERVER.INI, Application Server is closed by executing an EXIT CLEAR command.

2.5 Creating a Remote Server Connection

Procedure

1. From the File menu, choose New and then choose Remote Server to display the Create Remote Server dialog box.
2. In the Server Name text box, type the name of the server.
3. In the Service text box, type the service.
4. In the Port text box, specify the port that is used for TCP/IP communication with the server.
5. In the User name text box, type your username to log onto the server.
6. In the Password text box, type the password to log onto the server.
7. (Optional) To list all commands issued on the client, select Trace Application Server commands. In the Trace file name text box, specify the file name (and path) to contain the trace details. Then, select Detailed or Basic to specify the level of trace information to write for each command.
8. Click OK.

Notes:

This remote server information gets written to your lsserver.ini file.

The password is written to the lsserver.ini file in encrypted format.

If you want to be prompted to type your password every time you try to connect to a remote server rather than store the password encrypted in lsserver.ini, make sure the EncryptedPassword= value in the [remote server] section in the lsserver.ini file is either blank or has a question mark (?).

2.6 Running in Batch from the Command Line

Procedure

Enter the following command:

PASADMIN.EXE [parameters]

You can specify the following parameters in the command line:

Parameter	Action
-e <i>procname</i>	Executes a procedure in batch, where the procedure is <i>procname</i> .
-i <i>minutes</i>	Logs off if idle after the specified number of minutes.
-infile <i>name</i>	Uses the <i>name</i> file instead of LSSERVER.INI for the client. The file must be in the Windows directory.
-j <i>procname</i>	Runs the procedure in batch using the JOB command. Note: When specifying a path location of an external file, use single quotes around the full path. For example, -j 'c:\temp\test';ext.
-ns	No server log on, even if requested.
-p <i>password</i>	Password.
-pp	Displays the login dialog box, even if the user name and password have been specified.
-s <i>servername</i>	Name of server to access.
-serverinfile <i>name</i>	Uses the <i>name</i> file instead of LSSERVER.INI for the server process. This file is sent to TSERVER.EXE if Winsock is being used. This parameter cannot be used for the listener on UNIX, or for the Issagent on Windows NT, because the listener will have already been up and running before the server process is started.
-u <i>username</i>	Username to log in to Application Server.

Tip: The pasadmin.exe command line will return immediately if the user could not log in or the job is terminated immediately.

Examples

Example 1: This command line logs in the user as user ADMIN and the password BLUEBIRD, and then runs the procedure named TESTPROC using the JOB command:

```
PASADMIN.EXE -j TESTPROC -u ADMIN -p BLUEBIRD
```

Note: If you log in as ADMIN, it will use your default database. You can verify the default database by issuing a SUPERVISOR SHOW USER ADMIN command to see the default database (usually JUICE in READ mode). You may want to create a user that is used by the Batch scripts, and has admin rights, and uses a default database of *proclib* where *proclib* is the name of your procedures library database.

Example 2: This example runs procedures that are stored in an Application Server model. In this example, there is a database called PROCLIB that is attached to JUICE, and there is a procedure called "Test" that will be run in PROCLIB:

```
PASAdmin.exe -u admin -j test;PROCLIB
```

The procedure test just copies the procedure from TEST to TEST2... a very simple test.

```
copy proc test;proclib test2;proclib
```

Example 3: This example runs external Application Server procedures. The Application Server procedures are not stored in PROCLIB, but rather in c:\temp as flat files:

```
copy proc 'c:\temp\test';ext 'c:\temp\test2';ext  
PASAdmin.exe -u admin -j 'c:\temp\test';ext
```

2.7 Running Application Server Jobs in Batch from a UNIX Server

Sample UNIX shell scripts are provided to run Application Server jobs in a batch process. You can modify files named batch1, batchlss.ksh, and batchlss.csh to reflect your installation configuration. Almost all UNIX batch jobs are scheduled using the cron utility, which is available on all UNIX systems.

Procedure to display systems manual information about using the cron utility

Type the following at the UNIX prompt:

```
"man cron"
```

Procedure to submit Application Server jobs in a batch process

Enter a command in the following format:

```
nohup batchlss.csh &
```

In this command, the ampersand (&) runs the batchlss.csh script as a background process, and nohup ensures that batchlss.csh is protected from hangups and terminations.

Batchlss.ksh and batchlss.csh

Use the batchlss.csh shell script to run batch jobs using the C shell. The original batchlss.csh shell script defines all the required environment variables and then executes Application Server with the following commands:

```
$LSSHOME/lss <batch1 >&batch1.out
```

Use the batchlss.ksh shell script to run batch jobs using the Korn or Bourne shell. The original batchlss.ksh shell script defines all the required environment variables and then executes Application Server with the following commands:

exec \$LSSHOME/lss <batch1 >&batch1.out

These commands define the file batch1 as the source of all input that a user would normally enter from a terminal. Output and errors are written to the file batch1.out. You can create multiple copies of the batch scripts to run different jobs. You can edit the batch1 file to start with the Application Server user name and password, followed by the Application Server commands you want to run. For example, the original batch1 file looks like the following:

```
admin  
  
trace both batch1t;ext  
version  
show data  
status  
show variables  
trace off  
exit clear
```

This input file logs in to the ADMIN user name. The user name does not need a password so the next line is blank. The sample job then creates a trace file BATCH1T, which contains the results of the Application Server commands included in the batch job.

Note: You should include the commands to start and stop a trace in all batch jobs to produce a log file of the commands and their results. The final command must always be EXIT or EXIT CLEAR.

3 Application Server User Interface

3.1 Application Server Administrator Windows

Application Server Administrator has an Explorer window and a Trace window. Within the Explorer window, there are tabs and panes for working with your dimensional model and the IDQL commands.

Features

Trace window

The Trace window lists all the Application Server commands executed on the client system. The Trace window is hidden by default when you start Application Server.

Explorer window

The Explorer window is displayed when you first start Application Server.

It has these features:

Hierarchical pane

The hierarchical list pane contains the following hierarchical lists:

Dimensional Models - The Dimensional Models list contains information about all the dimensional models that exist in your MASTERDB. The following details are listed for each dimensional model: binary sets, dimensions, documents, logics, measures, procedures, reports, synonyms, time sets, and users.

To display details of dimensions, documents, logics, measures, procedures, reports, synonyms, time sets, and users, make sure the List tab is selected, and then click the appropriate icon in the Dimensional Models hierarchical list.

Security - The Security hierarchical list is displayed only to users with administrator authority, and contains details of users and User-Defined Hierarchies. It also contains security information for each of your dimensional models.

Link IDs - Click the Link IDs icon to display information about Link IDs in the List tab. A Link ID is a way to specify SQL connectivity between the client applications and a wide variety of database services. Link IDs simplify the connection process to the different servers, network protocols, and operating systems in your work environment. Link IDs support access to a number of databases. Your operating system determines which databases you can access with a Link ID.

Remote Servers - The Remote Servers icon is displayed only if you have performed a client/server installation. Click the icon to display information about remove servers in the List tab.

Schemas - The Schemas hierarchical list contains information about your Hybrid OLAP schemas, if implemented. The following information is listed for each schema:

The Partitions hierarchical list contains information about the schema usage; input, output and result dimensions; selected members; and User-Defined Hierarchies.

The Structure hierarchical list contains details of the dimensions, measures, and Fact tables for the schema.

The Subschemas hierarchical list contains details of the dimension views and schema views in the schema, and any dimensional models that you have created from the schema.

Data View

The Data View tab is displayed by default when you start Application Server, unless you are logged in as Supervisor.

When you start Application Server, the initial Data View is based on your Use database. Your current selections are maintained, with one exception — for non-across/down dimensions with more than one member selected, Application Server selects only the top member.

When you redisplay the Data View after entering commands in the IDQL tab, the Data View is updated to reflect the current model and selections.

If you have the Data View tab displayed when you switch to a different dimensional model in the hierarchical list pane, the view is refreshed immediately; otherwise, the view is updated when you next display the Data View tab. Therefore, if you want to perform small administrative tasks on several different dimensional models, operations will be faster if you display a different tab, or temporarily turn off the display of the Data View tab.

You can customize the appearance of the Data View tab by specifying gridlines, font properties, text and background colors. You can also specify colors to highlight high and low data values, and turn off the display of rows that contain only missing values.

List tab

The List tab displays different information according to your current selection in the hierarchical list. For example, if you select *Measures* in the hierarchical list, the List tab shows details of the measures in the currently selected dimensional model. If you select Link IDs in the hierarchical list, the List tab shows details of your default and user-created Link IDs.

You can right-click most items in the List tab to display a shortcut menu. The commands available depend on the type of item that is currently selected in the List tab.

You can sort a column by clicking the column heading.

IDQL tab

You can use the IDQL tab to execute Application Server commands directly in the Command window.

You use the Command window of the IDQL tab to execute commands in Application Server.

You can customize the appearance of the IDQL tab, by specifying the font to use. You can also clear the Output window after each command is executed.

You can specify the maximum amount of memory to use for the IDQL output. If the output in the window exceeds the specified memory, Application Server removes text from the start of the output until the remaining output fits into the buffer.

You can display the command history, and select a command to place in the Command window. You can also search the command history, and display the previous and next commands.

3.2 Executing an IDQL Command

Procedure

1. Click the IDQL tab.
2. In the Command window, type the IDQL command.
3. To execute the command, press Enter, or click the Execute button on the toolbar.

The Output window displays any output resulting from the IDQL command.

Notes:

You can use the command history to redisplay previously entered commands.

If you are running Application Server either in standalone mode or as client/server to a Windows server computer, you can click the Stop button on the toolbar to stop an executing IDQL command.

3.3 Syntax Conventions

Commands are shown in uppercase characters. For example:

CLEAR STATUS

Names that you must substitute with a value are shown in lowercase italic characters. In this example, you enter SAVE and then substitute an actual procedure name for *procedure*:

SAVE <procedure>

Brackets [] signify that the words inside them are optional. In the following example, you can enter RECOVER *setname* or RECOVER *settype setname*:

RECOVER [<settype>] <setname>

When a command shows a list of bracketed options, you can specify any number of options. In this example, you can enter SET PRINTER WIDTH 80 LENGTH 60 UPPER.

SET PRINTER [WIDTH <width>] [LENGTH <length>] [UPPER]

When options are enclosed in brackets and separated by a vertical line, it means that you can specify one option but you do not have to specify any options. In this example, you can enter READ, READ ADD, or READ SUBTRACT. (This is not the complete READ command.)

READ [ADD | SUBTRACT]

When options are enclosed in braces {} and separated by a vertical line |, it means that you must choose one option from between the brackets. In this example, you must enter CHECKPOINT with one option, either BACKUP, FREEZE, or UPDATE [*n*]. You cannot enter just CHECKPOINT, and you cannot enter CHECKPOINT with two options, such as CHECKPOINT FREEZE UPDATE.

CHECKPOINT {BACKUP | FREEZE | UPDATE [<n>]}

When a command contains a dotted list, you enter the command once and follow it with a list of statements. In the following example, you would enter WHILE *condition*, enter one or more statements, and then enter ENDWHILE.

WHILE condition

.

. **statements**

.

ENDWHILE

When a command has a list of options in braces {}, you enter the command with one option from the list of options. In this example, you can enter EXHIBIT ACROSS, or EXHIBIT DOWN, or EXHIBIT LOGIN, and so on. It is similar to the CHECKPOINT example, but is used for extensive lists that do not fit on a single line. (This is not the complete EXHIBIT command.)

EXHIBIT

```
{ ACROSS }
{ ADIMENSION [<attribute>] [BASIS | FULL]
{ [ALL] [SELECTED] MEMBERS <dimension>
[HIERARCHY <hierarchy>][<keyword1>][BOTH][POSITION] }
{ ATTRIBUTES [<dimension>] }
{ BASIS <attribute> [BOTH] }
```

When a command has a list of options in brackets[], you enter the command and any number of options. In this example, you can enter SET COLUMNS 5 INDENT 10 ENDSET. It is similar to the

SET PRINTER example, but it is used for extensive lists that do not fit on a single line. (This is not the complete SET-ENDSET command.)

```
SET[ACROSS <across>]
    [COLUMNS <n>]
    [DECIMALS <list>]
    [DOWN <down>]
    [INDENT <spaces>]
```

ENDSET

When a word is bracketed with ellipses, you can specify one or more values separated by commas. In this example, you would type LEVEL and then one or more field names separated by commas.

```
LEVEL <field> [...<field>]
```

When a word is plural and in lowercase letters, you can specify one or more values separated by commas. In the following example, you can enter PRINT with a list of set names to be printed, separated by commas:

```
PRINT [<settype>] <setnames>
```

Command separators

By default, Application Server uses a bar symbol (|) as a command separator. To change this, edit the following line in the LSSERVER.INI file:

```
CMDSEP = |
```

Take care when changing the command separator, as some characters can cause Application Server to function incorrectly. However, the bar (|) symbol, the exclamation point (!), and the tilde (~) do not clash with normally used characters in file names and the Application Server command language.

Set name conventions

Set names (such as `reports`) must start with an alphabetic or underscore characters(except dimensions, where the underscore is reserved). Subsequent characters can be alphanumeric or underscore.

Colons in variable names or labels

Do not use colons (:) in variable names or labels.

Reserved words

The Application Server commands and command options should not be used for user defined names because they are reserved words.

Reserved characters

There are five characters which Application Server uses as member encapsulation characters or "quote characters". The five characters are:

- ' single quotation mark
- " double quotation mark
- < less than sign
- > greater than sign
- ` apostrophe

These characters are all used in pairs with special characters, with the exception of the greater than (>) and less than (<) signs, which are used in opposition to each other. For example:

<Margin %>

"Margin %"

It is recommended that you avoid using reserved characters in member or variable names.

Also, serious problems can occur when trying to issue a complex select command if two or more reserved characters are used a single dimension member. For example:

"Bob's Six O'Clock Coffee Store's Margin"

Special characters

A special character is any character that is not alphanumeric or an underscore. When one or more of these characters is used in variable and dimension member names, the name must be enclosed in single quotation marks (' '). For example, the following statement is valid:

CREATE VARIABLE 'Margin %'

Set names cannot contain special characters and cannot begin with a number.

Variables cannot begin with an exclamation point (!).

3.4 Controlling Your View

3.4.1 Changing the Background Colors in the Data View

Procedure to change the background color of the headings

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. In the Headings group, click the Background button to display the Color dialog box.
4. Specify a basic or custom color, and then click OK to close the Color dialog box.
5. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

Procedure to change the background color in the data area

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. In the Data Area group, click the Background button to display the Color dialog box.
4. Specify a basic or custom color, and then click OK to close the Color dialog box.
5. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.2 Changing the Color of Text in the Data View

Procedure to change the color of the headings text

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. In the Headings group, click the Foreground button to display the Color dialog box.
4. Specify a basic or custom color, and then click OK to close the Color dialog box.
5. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

Procedure to change the color of the text in the data area

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. In the Data Area group, click the Foreground button to display the Color dialog box.
4. Specify a basic or custom color, and then click OK to close the Color dialog box.
5. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.3 Changing the Default Font Properties

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Fonts tab.
3. In the "Screen area" drop-down list, select a region of the screen.
4. To change the font name for the selected area, in the Font list box, select a different font.
5. To change the font size for the selected area, either type a new size in the Size text box, or select a different option in the Size list box.
6. Do one of the following:
 - Click OK to confirm your changes and close the Options dialog box.
 - Click Apply to apply your changes and keep the Options dialog box open.
 - Click Cancel to cancel your changes and close the Options dialog box.

3.4.4 Setting the Default Login Properties

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Login tab.
3. Specify the default login method to use when the Application Server program starts by doing one of the following:
 - To display the Application Server login dialog box when the Application Server program starts, select "Display a login dialog box at startup". This is the default value.
 - To log in automatically when the Application Server program starts, specify the following:
 - a. Select "Automatically login using these settings".
 - b. In the "User name" text box, enter an existing user name.
 - c. In the Password text box, enter the password for the specified user.
 - d. In the Server text box, enter the name of the server to connect to. If a remote server is specified, then specify a valid remote server connection.
4. Do one of the following:
 - Click OK to confirm your changes and close the Options dialog box.
 - Click Apply to apply your changes and keep the Options dialog box open.
 - Click Cancel to cancel your changes and close the Options dialog box.

Note: If you define a default login, the details are written to the Application Server initialization file, (the default initialization file name is LSSERVER.INI).

3.4.5 Clearing the IDQL Output After Each Command

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Command Window tab.
3. Do one of the following:
 - To clear the output window of the IDQL tab after each command, select "Clear IDQL output after each command".
 - To keep the output after each IDQL command, deselect "Clear IDQL output after each command".
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.6 Confirming Drag and Drop Operations

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Confirmation tab.
3. Do one of the following:
 - To prompt the user for confirmation when executing a drag and drop operation, select "Confirm drag and drop operations".
 - To execute drag and drop operations without user confirmation, deselect "Confirm drag and drop operations".
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.7 Confirming Item Deletion

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Confirmation tab.
3. Do one of the following:
 - To prompt the user for confirmation before deleting an item, select "Confirm before deleting an object".
 - To delete items without user confirmation, deselect "Confirm before deleting an object".
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.8 Confirming Exit from Application Server

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Confirmation tab.
3. Do one of the following:
 - To prompt the user to save the Work database before exiting Application Server, select "Confirm before exiting Application Server".
 - To exit from Application Server without prompting the user to save the Work database, deselect "Confirm before exiting Application Server".

4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.9 Displaying or Hiding Gridlines in the Data View

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. Do one of the following:
 - To display gridlines in the Data View, select "Show gridlines".
 - To hide the gridlines in the Data View, deselect "Show gridlines".
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.10 Suppressing the Display of Missing Rows

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Data View tab.
3. Do one of the following:
 - To suppress the display of rows that contain only missing values in the Data View, select "Suppress missing rows".
 - To display rows that contain only missing values in the Data View, deselect "Suppress missing rows".
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

3.4.11 Displaying or Hiding the Status Bar

From the View menu, choose Status Bar.

Note: The Status Bar command is a toggle, that is, you choose the command to turn it on, and choose it again to turn it off. A check mark next to the menu command indicates that it is selected.

3.4.12 Displaying or Hiding the Toolbar

From the View menu, choose Toolbar.

Note: The Toolbar command is a toggle, that is, you choose the command to turn it on, and choose it again to turn it off. A check mark next to the menu command indicates that it is selected.

3.4.13 Displaying the Command History

Procedure to display the command history

Press F7 to display the command history, where you can select a command to place in the command window.

Procedure to search the command history

Do one of the following:

Type part of a command string in the command window, and then press F8 to search the command history for a command that begins with the specified characters.

Press F8 again to display the next command that begins with the specified characters.

Press F9 to display a dialog box where you can select a command from the history based on its number.

Procedure to display the previous command

Press the CTRL+UP arrow key, or click the Previous button on the toolbar.

Procedure to display the next command

Press the CTRL+DOWN arrow key, or click the Next button on the toolbar.

3.4.14 Specifying the Columns to Display in the List Tab

Procedure

1. Make sure the List tab is selected.
2. Right-click the List tab to display a shortcut menu, and then choose Select Columns to display the Select Columns dialog box.
3. You can do the following:
 - To display a column in the List tab, drag the field from the "Available columns" list to the "Selected columns" list. Or select the field and then click the Add button.

Note: New items are added to the "Selected columns" list below the currently selected item in the list.
 - To add all available columns to the "Selected columns" list, click the Add All button.
 - To remove a column in the List tab, drag the field from the "Selected columns" list to the "Available columns" list. Or select the field and then click the Remove button.

Note: You cannot remove the first field from the "Selected columns" list.
 - To remove all available columns (apart from the first field in the list) from the "Selected columns" list, click the Remove All button.
4. Click OK.

3.4.15 Specifying the Maximum Memory for the IDQL Output and SQL Command Windows

Procedure

1. From the View menu, choose Options to display the Options dialog box.
2. Click the Command Window tab.
3. In the "Maximum memory to use for command output " text box, enter a value in Kb.
4. Do one of the following:
 - Click OK to confirm your changes and close the dialog box.
 - Click Apply to apply your changes and keep the dialog box open.
 - Click Cancel to cancel your changes and close the dialog box.

Notes:

The specified value applies to each window separately. The default value for each window is 256 Kb.

If the output in the window exceeds the specified memory, Application Server removes text from the start of the output until the remaining output fits into the buffer.

The application title bar, menu bar, and toolbar are hidden in full screen mode, in order to maximize the size of the available working area in the Explorer window.

3.4.16 Switching to Full Screen Mode

From the View menu, choose Full Screen.

Note: The Full Screen command is a toggle; that is, you choose the command to turn it on, and choose it again to turn it off. A check mark next to the menu command indicates that it is selected.

3.5 Working with Items

3.5.1 Creating a Dimension

Dimensions are components of a dimensional model. When building a dimensional model, create structural dimensions to describe an area of interest within your business, or attribute dimensions to describe a characteristic of a structural dimension.

Dimensions consist of members organized into a hierarchical structure. A dimension's hierarchical structure is the road map for drilling up and down on information.

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Dimension.
3. In the Create Dimension dialog box, specify a unique name for the new dimension.
4. Click OK to display the Dimension editor, where you can create a new dimension.
5. (Optional) To update the dimension details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a dimension if you are using the selected dimensional model in Read-only mode.

3.5.2 Creating a Document

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Document to display a dialog box, where you can specify a unique name for the new document.
3. Click OK to display the Document editor, where you can create a new document.
4. Close the Document editor, and save the document.
5. (Optional) To update the document details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a document if you are using the selected dimensional model in Read-only mode.

3.5.3 Creating a Logic

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Logic to display a dialog box, where you can specify a unique name for the new logic.
3. Click OK to display the Logic editor, where you can create a new logic.
4. Close the Logic editor, and save the logic.
5. (Optional) To update the logic details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a logic if you are using the selected dimensional model in Read-only mode.

3.5.4 Creating a Procedure

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Procedure to display a dialog box, where you can specify a unique name for the new procedure.
3. Click OK to display the Procedure editor, where you can create a new procedure.
4. Close the Procedure editor, and save the procedure.
5. (Optional) To update the procedure details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a procedure if you are using the selected dimensional model in Read-only mode.

3.5.5 Creating a Report

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Report to display a dialog box, where you can specify a unique name for the new report.
3. Click OK to display the Report editor, where you can create a new report.
4. Close the Report editor, and save the report.
5. (Optional) To update the report details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a report if you are using the selected dimensional model in Read-only mode.

3.5.6 Creating a Synonym

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Synonym to display a dialog box, where you can specify a unique name for the new synonym.
3. Click OK to display the Synonym editor, where you can create a new synonym.
4. Close the Synonym editor, and save the synonym.
5. (Optional) To update the synonym details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a synonym if you are using the selected dimensional model in Read-only mode.

3.5.7 Creating a Time Set

Procedure

1. Make sure that you have selected a dimensional model in the hierarchical list pane.
2. From the File menu, choose New, and then choose Time Set to display a dialog box, where you can specify a unique name for the new time set.
3. Click OK to display the Time Set editor, where you can create a new time set.
4. Close the Time Set editor, and save the time set.
5. (Optional) To update the time set details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Notes:

You may need to change your access mode, as you cannot create a time set if you are using the selected dimensional model in Read-only mode.

3.5.8 Creating a User

Procedure

1. From the File menu, choose New, and then choose User to display the Create User dialog box.
2. Enter the user details as follows:
 - a. In the "User name" text box, enter a unique user name of up to 24 characters. The first character must be alphabetic — the remaining 23 characters can be alphanumeric.
 - b. (Optional) In the Password text box, enter a password of up to 12 alphanumeric characters.
3. Enter the default model details for the user as follows:
 - a. In the "Default model name" drop-down list, select the default model for the user.
 - b. In the "Default access mode" drop-down list, specify how the user will access the model. The default value is Exclusive.
4. Specify the Work model details as follows:
 - a. In the "Work model name" text box, specify a name of up to eight characters for the user's Work database.
 - b. In the "Number of buffers" text box, specify the number of buffers (between 20 and 64,000) to allocate to the Work database. The default value is 2000.
 - c. In the "Number of blocks" text box, specify the number of blocks (the maximum number you can specify is disk and partition dependent) to allocate to the Work database. The default value is 2000.
 - d. In the "Block size" drop-down list, specify the Work database block size. The default value is 8K.

Note: For more information about database size, see Determining buffers setting.
5. Specify the user login details as follows:
 - a. In the "Security level" drop-down list, select the authority level for the user.

Note: Supervisor level has access to all Application Server commands.
 - b. In the "Account status" drop-down list, specify whether the user account is Enabled (default value), or Disabled.
 - c. In the "Number of logins" text box, enter a value between 1 and 15 to specify the number of times the user can log in concurrently to Application Server.

Note: Values greater than 1 allow multiple users to access Application Server with the same user name.
 - d. In the "Expiration date" text box, specify the date when the user account expires. The default value is Never.
6. Click OK to close the dialog box.

Notes:

The user is added to MASTERDB, and appears in the Users list in the hierarchical list pane.

You can right-click a user in the List tab to display a shortcut menu where you can log the user out, or enable a user whose record is displayed.

3.5.9 Creating a User Group

Procedure

1. From the File menu, choose New, and then choose User Group to display a dialog box, where you can specify a unique name for the new user group.
2. Click OK Creating the user group.

The user group is added to the User Groups list in the hierarchical list pane — you can now add users to the group.

3.5.10 Copying an Item to the Clipboard

Procedure

1. Select the item you want to copy to the Clipboard.
2. Do one of the following:
 - From the Edit menu, choose Copy.
 - Display the shortcut menu for the item, and then choose Copy.
 - Click the Copy button on the toolbar.

Note: Copy makes a copy of the selection and saves it on the Clipboard, overwriting whatever is currently on the Clipboard. The selection remains on the screen.

3.5.11 Copying an Item to a Different Dimensional Model

Procedure

1. In the List tab, select the item you want to copy.
2. Drag the item to a different dimensional model in the hierarchical list pane.
3. If you have "Confirm drag and drop operations" selected in the Options dialog box, you will be prompted to confirm the operation. Do one of the following:
 - Click Yes to copy the selected item.
 - Click No to cancel the copy operation.

3.5.12 Cutting Text to the Clipboard

Procedure

1. Select the text you want to cut to the Clipboard.
2. Do one of the following:
 - From the Edit menu, choose Cut.
 - Display the shortcut menu, and then choose Cut.
 - Click the Cut button on the toolbar.

Note: Cut removes the selected text from the screen, but saves it on the Clipboard, overwriting whatever is currently on the Clipboard. You can use Paste to place the selection back on the screen.

3.5.13 Deleting an Item

Procedure

1. In the hierarchical list pane, select the class of item you want to delete, and then click the List tab.
2. In the List tab, select the item you want to delete.
3. Do one of the following:
 - From the File menu, choose Delete.
 - Right-click the item in the List tab to display a shortcut menu, and then choose delete.
 - Press the Delete key.
4. If you have "Confirm before deleting an object" selected in the Options dialog box, you will be prompted to confirm the deletion. Do one of the following:
 - Click Yes to delete the selected item.
 - Click Cancel to cancel the deletion.

Note: To delete items from a dimensional model, you must be using the model in Shared or Exclusive mode. To delete a dimension, measure, or attribute, you must be in Exclusive mode.

3.5.14 Displaying a Shortcut Menu

Right-click in the Explorer, Data View, List, or IDQL tabs.

The commands displayed depend on the item that is selected when you right-click.

3.5.15 Displaying the Properties of an Item

Display the shortcut menu for the item, and then choose Properties to display a dialog box that shows the properties of the selected item.

3.5.16 Editing an Item

Display the shortcut menu for the item, and then choose Edit to display the appropriate editor for the selected item.

Note: If the Use database is attached in Read-only mode, the set is open for viewing only. You can edit dimensions only when the database is open in Exclusive mode. For more information, see Changing your access mode.

3.5.17 Pasting from the Clipboard

From the Edit menu, choose Paste. Or, click the Paste button on the toolbar.

Notes:

You use Paste to place the contents of the Clipboard on the sheet.

You can only paste back the item(s) that was last cut or copied to the Clipboard.

3.5.18 Saving an Item

Procedure to save an item

1. From the File menu, choose Save to display the File Save As dialog box. Or, click the Save button on the toolbar.
2. In the "Save in" drop-down list, specify a drive and location to save the file to.
3. Type a name for the file in the "File name" text box, or double-click an existing name in the file window to overwrite it with the new file.
4. Make sure the correct type is selected in the "Save as type" list box.
5. Click Save.

Procedure to save an item with a new name

1. From the File menu, choose Save As to display the File Save As dialog box.
2. In the "Save in" drop-down list, specify a location to save the file to.
3. In the "File name" text box, type a name for the file. Or, double-click an existing name in the file list to overwrite it with the current file.
4. Make sure the correct type is selected in the "Save as type" drop-down list.
5. Click Save.

3.5.19 Selecting an Item

Procedure to select an item

1. Make sure the Explorer window is displayed.
2. In the hierarchical list pane, click the plus symbol (+) beside the item name to expand the list.
3. In the list, click the item name or the icon for the item class (for example, Documents, Procedures, Reports) that you want to select.
4. Click the List tab to see details of the selected item, or a list of items of the selected item class (for example, a list of documents in the current dimensional model).

Procedure to deselect an item

Click a different item, or an empty area of the screen.

3.5.20 Printing an Item

You can print in the SQL Command window, the Trace window, the Application Server editors, the Explorer window when the IDQL tab is selected in the right pane, and the command window of the IDQL tab.

Procedure

1. From the File menu, choose Print to display the Print dialog box.
2. Select a printer from the list of installed printers.
3. (Optional) Click Properties to open a dialog box for the selected printer, in which you can specify a paper tray, resolution, and so on.
4. (Optional) Select "Print to file", and then click OK to display the Print to File dialog box.
Type the file name and then click OK. You can send the file to the printer using the DOS COPY command.

5. (Optional) To print more than one copy, enter the number of copies in the "Number of copies" text box. Or, use the spin buttons.
6. Click OK.

You can print in the SQL Command window, the Trace window, the Application Server editors, and the Explorer window when the IDQL tab is selected in the right pane.

3.5.21 Changing Printer Default Setup Settings

Procedure

1. From the File menu, choose Print Setup to display the Print Setup dialog box.
2. Select a printer from the list of installed printers.
3. (Optional) Click Properties to open a dialog box for the selected printer, in which you can specify paper trays, resolution, and so on.
4. Make selections for paper source and size, and orientation.
5. Click OK.

The changes become the default settings the next time you print.

3.5.22 Displaying an Item Before Printing

You can display items before printing in the SQL Command window, the Trace window, the Explorer window when the IDQL tab is selected in the right pane, and the command window of the IDQL tab.

From the File menu, choose Print Preview.

Note: To cancel Print Preview, press Esc.

3.6 Working with Data in the Data View Tab

3.6.1 Adding a Dimension to the Data View Across, Down and Page Lists

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. In the hierarchical list pane, expand the Dimensions list for the currently selected model.
3. Select a dimension in the list in the hierarchical list pane, and then drag it to the Across, Down, or Page lists of the Data View tab.

If you have "Confirm drag and drop operations" selected in the Options dialog box, you will be prompted to confirm the operation.

Notes:

You can add a total of up to 14 dimensions to the Across, Down, and Page lists, including the mandatory Time and Measures dimensions.

You can also change the Across, Down and Page lists by dragging a dimension from another list. The Across and Down lists must contain at least one dimension.

3.6.2 Changing the Across, Down and Page Lists

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Change the Across, Down or Page lists by dragging dimensions from another list, for example from the Across list to the Down list.
3. Select a dimension in the list in the hierarchical list pane, and then drag it to the Across, Down, or Page lists of the Data View tab.

If you have "Confirm drag and drop operations" selected in the Options dialog box, you will be prompted to confirm the operation.

Notes:

The Across and Down lists must contain at least one dimension.

You can also add a dimension from the list in the hierarchical list pane. You can add a total of up to 14 dimensions to the Across, Down, and Page lists, including the mandatory Time and Measures dimensions.

3.6.3 Swapping the Dimension Display Direction

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Click the Swap Across Down button on the toolbar.

The contents of the Across and Down lists are switched simultaneously.

3.6.4 Displaying a Different Page

Procedure to display the next page

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Click the Next button on the toolbar.

Procedure to display a different page

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Click the Goto Page button on the toolbar to display the Go To Page dialog box.
3. In the list box, select the page you want to display, and then click OK.

3.6.5 Removing a Dimension from the View

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. To remove a dimension from the Across, Down, or Page lists, drag the dimension to the top left cell of the grid.

3.6.6 Changing the Drill Direction

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Click the Drill Direction button on the toolbar to reverse the current drill direction.

3.6.7 Changing the Periodicity

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Do one of the following:
 - Click the Calendar button on the toolbar.
 - Right-click in the view to display a shortcut menu, and then choose Calendar to display the Calendar dialog box.
3. Make sure the Periodicity tab is selected.
4. In the Periodicity list box, select a different option, and then click OK to close the Calendar dialog box.

The view of the dimensional model data is updated to reflect the changed periodicity.

3.6.8 Changing the Time Template

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Do one of the following:
 - Click the Calendar button on the toolbar.
 - Right-click in the view to display a shortcut menu, and then choose Calendar to display the Calendar dialog box.
3. Click the Templates tab.
4. (Optional) In the "Template type" drop-down list, select a different template type.
The Templates list box is updated to show time templates based on the selected type.
5. In the Templates list box, select a time template to apply to the dimensional model data.

6. Click OK to close the Calendar dialog box.

The view of the dimensional model data is updated to reflect the changed time template.

3.6.9 Displaying the Dimensional Selector

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Do one of the following:
 - Select a dimension in the view, and then click the Selector button on the toolbar.
 - Display the shortcut menu, and then choose Selector.
 - Display the Viewer, and then double-click a dimension.

3.6.10 Adding a Member to a Selected Dimension

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Display the Dimensional Selector dialog box.
3. In the Hierarchy list box, click the plus symbol (+) beside a dimension to expand the member list, or click Expand All to display expanded member lists for all dimensions.
4. Add one or more members to the dimension as follows:
 - To add a member, select it in the Hierarchy list box, and then click Add to add it to the "Selected members" list. Or, double-click the member in the Hierarchy list box.
 - To add all the members at once, click Mark All, and then click Add to add them to the "Selected members" list box. You can add 1024 members at a time. Repeat this step to add more than 1024 members.
5. (Optional) To display the selected members in the order in which they appear in the "Selected members" list box at the time you click OK, select "Order as selected".
6. (Optional) Click the Find tab to build a list of members that match a specified search string.
You can add the found members to, or remove them from, the "Selected members" list box.
7. (Optional) Click the Attributes tab to work with the attributes of the selected dimension.
You can use attributes to add members to the "Selected members" list box, or to restrict the "Selected members" list box to display only members that have specific attributes.
8. Click OK to close the Dimensional Selector dialog box, and return to the view.

3.6.11 Removing a Member from a Selected Dimension

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Display the Dimensional Selector dialog box for a selected dimension.
3. In the Hierarchy list box, click the plus symbol (+) beside a dimension to expand the member list, or click Expand All to display expanded member lists for all dimensions.
4. In the "Selected members list box", select the member you want to remove, and then click Remove.
5. Click OK to close the Dimensional Selector dialog box.

3.6.12 Removing All Members from a Selected Dimension

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Display the Dimensional Selector dialog box for a selected dimension.
3. Click Delete All, and then click OK to return to the Data View tab.

3.6.13 Searching for Dimension Members

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Display the Dimensional Selector dialog box for a selected dimension.
3. Click the Find Members tab.
4. In the "Search expression" text box, type a search string.

Notes:

- By default, or if you have used the SET LONG command in Application Server to display long dimension member names (labels), you can search using a wildcard character (*), a long or short name, or part of a long or short name followed by a wildcard character (for example Court*).
 - If you have used the SET SHORT command in Application Server to display short dimension member names, you can search using a wildcard character (*), a short name, or part of a short name followed by a wildcard character (for example UPC1*).
5. Do one of the following:
 - To replace the current contents of the "Search result" list box with the results of the new search, click Find.

- To add the results of the new search to the current contents of the "Search results" list box, click Find Also.
6. You can add the found members to, or remove them from, the "Selected members" list box:
 - To add the found members to the "Selected members" list, click Add to Selection.
 - To remove the found members from the "Selected members" list, click Add to Selection.
 7. Click OK to close the Dimensional Selector dialog box.

3.6.14 Displaying the Properties of a Dimension

Do one of the following to display the Dimension Properties dialog box:

In the hierarchical list pane, select a dimension, and then from the File menu, choose Properties.

In the hierarchical list pane, right-click a dimension name to display a shortcut menu, and then choose Properties.

In the List tab, double-click a dimension name.

In the List tab, right-click a dimension name to display a shortcut menu, and then choose Properties.

You can specify a high and low value for data in the Data View, so that when the data in the cells falls above or below the high or low value, the cell contents are displayed in the specified color.

3.6.15 Specifying Colors for High and Low Data Values

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Right-click the data to display a shortcut menu, and then choose "High and Low Values" to display the High and Low Values dialog box.
3. Deselect "Do not apply high and low coloring".
4. In the "High value" and "Low value" text boxes, enter high and low exception values.
5. To define how data exceeding the specified high value appears:
 - a. Click the button next to the "High value" text box to display the Color dialog box.
 - b. Select a background color to apply to data that exceeds the high value.
 - c. Click OK.
6. To define how data falling below the specified low value appears:
 - a. Click the button next to the "Low value" text box to display the Color dialog box.
 - b. Select a background color to apply to data that falls below the low value.
 - c. Click OK.
7. Click OK to close the High and Low Values dialog box.

Exceptions data in the Data View is shown in the specified colors.

3.6.16 Switching to another Dimensional Model

Procedure

1. Make sure the Data View tab is displayed.
2. In the hierarchical list pane, select a different dimensional model.

The Data View displays the current selections for the newly selected model.

Note: Only the top member is selected for non-across/down dimensions.

3.6.17 Displaying the Calendar

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Do one of the following:
 - Click the Calendar button on the toolbar.
 - Display the shortcut menu, and then choose Calendar.
 - Display the Viewer, and then double-click Time.

3.6.18 Displaying the Viewer

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. Display the shortcut menu, and then choose Viewer.

3.6.19 Saving a View

Procedure

1. Make sure the Data View tab is displayed for the dimensional model currently selected in the hierarchical list pane.
2. From the File menu, choose Save As to display the File Save As dialog box.
3. In the "Save in" drop-down list, specify a database to save the view to.

Note: If the database is not currently attached, it is attached as shared. You cannot save the view to your Use database unless the database is already attached in Shared or Exclusive mode.

4. Type a name for the view in the "View name" text box, or double-click an existing name in the file window to overwrite it.
5. Click Save.
6. Once you have saved a view, you can choose Save or click the Save button on the toolbar to save a new view to the same View name.

3.6.20 Opening a Saved View

Procedure

1. Make sure the Data View tab is displayed for the dimensional model that contains the saved view.
2. From the File menu, choose Open and then choose View to display the Open dialog box. Or, click the Open View button on the toolbar.
3. In the "Look in" drop-down list, select the database that contains the saved view, and then select the required view.
4. Click Open to display the saved view in the Data View tab.

3.7 Working with Link IDs

3.7.1 What Is a Link ID?

A Link ID template is a unique way for you to specify SQL connectivity between the client applications and a wide variety of database services. Link IDs simplify the connection process to the different servers, network protocols, and operating systems in your work environment.

You can construct a dictionary of connection parameters to database services by defining connections and saving them under unique names. You create and edit existing connection names using the Application Server.

Link IDs support access to a number of databases. Your operating system determines which databases you can access with a Link ID.

On Windows, a Link ID can be used to access an OLE DB or OLE DB for OLAP (ODBO) provider as a data source. OLE DB is a means to support a wider variety of data sources that are non-relational databases, such as object databases and spreadsheets that do not necessarily implement SQL.

ODBO is an extension to OLE DB that provides access to OLAP data sources that implement the MDX command language (Multi Dimensional eXpressions). You should be able to access any OLE DB or ODBO provider that you have installed on your system. If a data source is relational and supports SQL and you have a native driver such as OCI for Oracle, or an ODBC driver, these would be a better choice than going through OLE DB.

3.7.2 Creating a New Link ID

Procedure

1. From the File menu, choose New, and then choose Link ID to display the Create Link ID dialog box.
2. Select the database type you want to use from the list box, and then click OK to display the Link ID Properties dialog box.
3. In the Link ID Properties dialog box, define the connection information. The information required depends upon the database you are using.
4. Click Test to test the connection information you defined in the Editing Link ID dialog box.
If all parameters are correct a confirmation message appears.
5. Click OK to acknowledge the message, and then click OK again to close the Link ID Properties dialog box.

3.7.3 Creating a Link ID for an OLE DB or ODBO Provider

Procedure

1. From the File menu, choose New, and then choose Link ID to display the Create Link ID dialog box.
2. Select the OLE DB provider and click OK to display the Data Link Properties dialog box.
3. Select the OLEDB provider from the list and click Next to move to the Connection tab.
4. Define the connection information. The information required depends upon the database you are using.
5. Click Test Connection to test the connection information. If all parameters are correct a confirmation message appears.
6. Click OK to acknowledge the message.
7. Click OK to close the dialog box. The Link ID Properties dialog box is displayed.
8. Enter a name for the Link I. **Note:** For connections to the SAP NetWeaver BI ODBO provider, you must also provide a Client Number and a Client Language.
9. Click OK.

3.7.4 Deleting a Link ID

Procedure

1. Make sure the List tab is displayed.
2. On the Explorer tab, expand the Link IDs folder to display a list of existing Link IDs in the List tab.
3. In the List tab, right-click the Link ID you want to delete.
4. When the shortcut menu appears, choose Delete to display a confirmation dialog box.
5. Do one of the following:
 - To delete the selected Link ID, click Yes.
 - To cancel the deletion, click No.

3.7.5 Viewing or Editing an Existing Link ID

Procedure

1. Make sure the List tab is displayed.
2. On the Explorer tab, expand the Link IDs folder to display a list of existing Link IDs in the List tab.
3. In the List tab, right-click the Link ID you want to edit.
4. When the shortcut menu appears, choose Edit to display the Link ID Properties dialog box.
5. Modify the information you want in the text boxes.
6. Click Test to test the connection information you defined.
If all parameters are correct a confirmation message appears.

- Click OK to acknowledge the message, and then click OK again to close the Link ID Properties dialog box.

3.7.6 Testing Settings for an Existing Link ID

Procedure

- Make sure the List tab is displayed.
- On the Explorer tab, expand the Link IDs folder to display a list of existing Link IDs in the List tab.
- In the List tab, right-click the Link ID you want to test.
- When the shortcut menu appears, choose Properties to display the Link ID Properties dialog box.
- Click Test to test the connection information defined in the Link ID Properties dialog box.
If all parameters are correct a confirmation message appears.
- Click OK to acknowledge the message, and then click OK again to close the Link ID Properties dialog box.

3.7.7 Link ID Templates

3.7.7.1 Windows

3.7.7.1.1 Oracle (OCI)

This configuration provides access to Oracle through the Oracle call interface (OCI). Link executes the SQL statements by sending them to the Oracle database system. Any SQL statement supported by the database system can be executed. See your Oracle documentation for more information.

Parameter	Description
User name	You must supply an Oracle user name to connect.
Password	You must supply an Oracle password to connect.
Server	The SQL*Net connect string designating the server computer and database to be accessed for remote Oracle databases. The information required varies depending on the SQL*Net driver you are using. See your Oracle SQL*Net documentation for more information. You must enter a server name, even when Application Server and Oracle are on the same system.
Library	The Oracle library associated with your version of Oracle's SQL*Net. The required value varies depending on the version of the Oracle client software installed. If the default value of OCIW32.DLL fails, try OCI.DLL. The library parameter is not used for UNIX; you do not need to change this value.

System requirements

Link requires that you use the Oracle SQL*Net product to provide access to remote Oracle databases. The Oracle SQL*Net component must have been previously installed.

3.7.7.1.2 SQL Server (DBLib)

This configuration provides access to the Microsoft SQL Server through the Microsoft DB-Library interface. Link executes the SQL statements by sending them to the SQL Server database system.

Any SQL statements supported by the database system can be executed from Link. See your SQL Server documentation for more information.

Parameter	Description
User name	You must supply a SQL Server user name to connect.
Password	You must supply a SQL Server password to connect.
Server	Server containing the SQL Server database tables that you want to access.
Database	(optional) Name of the database to connect to.

System requirements

You must have the appropriate Net-Library and DB-Library installed to gain access to Microsoft SQL Server databases.

For Windows NT, NTWDBLIB.DLL is the DB-Library. The Net-Library you need depends on the network protocol used to connect to the SQL Server. For example, Named Pipes requires DBNMPNTW.DLL, while TCP/IP requires DBMSSOCN.DLL. Contact your Microsoft SQL Server vendor for the appropriate DB-Library and Net-Library.

Select this option if you have Microsoft DBLib/NetLib client software installed.

3.7.7.2 UNIX

3.7.7.2.1 Oracle (OCI)

This configuration provides access to Oracle through the Oracle call interface (OCI). Link executes the SQL statements by sending them to the Oracle database system. Any SQL statement supported by the database system can be executed. See your Oracle documentation for more information.

Parameter	Description
User name	You must supply an Oracle user name to connect.
Password	You must supply an Oracle password to connect.
Server	The SQL*Net connect string designating the server computer and database to be accessed for remote Oracle databases. The information required varies depending on the SQL*Net driver you are using. See your SQL*Net documentation for more information. You must enter a server name, even when Application Server and Oracle are on the same system.
Library	The Oracle library associated with your version of Oracle's SQL*Net. The required value varies depending on the version of the Oracle client software installed. If the default value of OCIW32.DLL fails, try OCI.DLL. The library parameter is not used for UNIX; you do not need to change this value.

System requirements

Link requires that you use the Oracle SQL*Net product to provide access to remote Oracle databases. The Oracle SQL*Net component must have been previously installed.

3.7.7.2.2 DB2 Universal DB (CLI)

The DB2 configuration provides access to IBM DB2 through the DB2 native interface called the DB2 Call Level Interface (CLI). The CLI is based on Microsoft ODBC and the ISO CLI.

Use your Link ID to allow Application Server to connect to a DB2 database, execute SQL commands and call CLI and DB2 APIs to perform actions on that database. You cannot issue CLP (Command Link Processor) commands or DB2 commands from within the Access LSLink subsystem."

Working with Link IDs

Parameter	Description
Link ID	Specifies the name to call this Link ID.
User name	(Optional) Specifies the user name for the operating system for authentication purposes. If you do not enter a user name and password, you will connect with the current operating system username and password. If the DB2 database to which you are connecting is a remote one on another server machine, you should supply the username on the server machine rather than your username on the local machine.
Password	(Optional) Specify the password for the operating system for authentication purposes. If you do not enter a user name and password, you will connect with the current operating system username and password. If the DB2 database to which you are connecting is a remote one on another server machine, you should supply the password for your username on the server machine rather than your password on the local machine.
Database	Name of the database with which to establish a connection.
Database Alias	Alternative name for the database, used for cataloguing purposes. Note: When you create a database on the server, the database alias is catalogued on the server with the same name as the database unless you specify otherwise. If you specify an alternate name, use the alternate name when specifying the database on the client. On WinX, if you also have an ODBC dataset name for that database, you can also refer to it by that name too since the connection mechanism is an ODBC function whether you are connecting via ODBC or the CLI. For example, if you create a DB2 database called PDE, a default alias of PDE, and an ODBC system DSN called PDEDB2, you can use either PDE or PDEDB2 as the database name in the Link dialog box.

System requirements

DB2 is accessible either through ODBC or through DB2's own native interface the DB2 Call Level Interface (CLI). The CLI is based on Microsoft ODBC and the ISO CLI. DB2 also supplies extra DB2-specific APIs for administrative and system functions which are not part of the CLI.

It is recommended that you always use the CLI interface rather than ODBC, since the CLI interface also uses the extra DB2-specific APIs not supported in ODBC, and the system will run more efficiently when it can use those extra APIs (for example the RUNSTATS API).

On the system designated to host the warehouse, the DB2 Database Administrator must do the following before creating a link ID:

1. Install DB2.
2. Create and start a DB2 instance.
3. Create the DB2 database.
4. Create DB2 tablespaces for the dimensional models.
5. Create a USER TEMPORARY TABLESPACE for db2 temporary tables created with DECLARE GLOBAL TEMPORARY TABLE.
6. Grant the USE privilege for the tablespaces.
7. If Application Server is on a separate system from the database, install DB2 client on the system where Application Server will reside. You must also catalog your remote DB2 databases on your client machine using the DB2 CLP command CATALOG DATABASE command.

3.8 Users, Security and Database Access

3.8.1 About MASTERDB and Default User Names

Application Server uses the MASTERDB file to store information about users and dimensional models. This database is the repository of all the meta data regarding Application Server databases and who can use them. To use a dimensional model, it must be added to MASTERDB. Only users with access to the Supervisor subsystem can update MASTERDB.

There are three default user names that already exist in MASTERDB:

admin

super

supervisor

If you want to add multiple users to a group, use the User Group Properties dialog box.

3.8.2 Adding a User to a User Group

Procedure to use the User Group Properties dialog box

1. In the hierarchical list pane, right-click the required user group to display a shortcut menu, and then choose Properties to display the User Group Properties dialog box.
2. Do one of the following:
 - To add a single user to the user group, select the user in the left pane of the dialog box, and then click the Add button.
 - Click the Add All button to add all available users to the user group.

Procedure to drag and drop users to add them to a user group

1. In the hierarchical list pane, expand the Security list, and then expand the User Groups list.
2. Do the following:
 - a. Make sure the List tab is displayed, and then click the Users folder to display a list of the existing users.
 - b. Drag the required user from the List tab to the required User Group in the hierarchical list pane.
 - c. If you have "Confirm drag and drop operations" selected in the Options dialog box, you will be prompted to confirm the addition of the user to the group. Do one of the following:
 - Click Yes to add the user to the group.
 - Click No to cancel the operation.
3. (Optional) In the hierarchical list pane, click the name of the user group to display an updated list of users in the List tab.

You can create users only if you have administrator authority.

3.8.3 Creating a User

Procedure

1. From the File menu, choose New, and then choose User to display the Create User dialog box.

2. Enter the user details as follows:
 - a. In the "User name" text box, enter a unique user name of up to 24 characters. The first character must be alphabetic — the remaining 23 characters can be alphanumeric.
 - b. (Optional) In the Password text box, enter a password of up to 12 alphanumeric characters.
3. Enter the default model details for the user as follows:
 - a. In the "Default model name" drop-down list, select the default model for the user.
 - b. In the "Default access mode" drop-down list, specify how the user will access the model. The default value is Exclusive.
4. Specify the Work model details as follows:
 - a. In the "Work model name" text box, specify a name of up to eight characters for the user's Work database.
 - b. In the "Number of buffers" text box, specify the number of buffers (between 20 and 64,000) to allocate to the Work database. The default value is 2000.
 - c. In the "Number of blocks" text box, specify the number of blocks (the maximum number you can specify is disk and partition dependent) to allocate to the Work database. The default value is 2000.
 - d. In the "Block size" drop-down list, specify the Work database block size. The default value is 8K.

Note: For more information about database size, see Determining buffers setting.
5. Specify the user login details as follows:
 - a. In the "Security level" drop-down list, select the authority level for the user.

Note: Supervisor level has access to all Application Server commands.
 - b. In the "Account status" drop-down list, specify whether the user account is Enabled (default value), or Disabled.
 - c. In the "Number of logins" text box, enter a value between 1 and 15 to specify the number of times the user can log in concurrently to Application Server.

Note: Values greater than 1 allow multiple users to access Application Server with the same user name.
 - d. In the "Expiration date" text box, specify the date when the user account expires. The default value is Never.
6. Click OK to close the dialog box.

Notes:

The user is added to MASTERDB, and appears in the Users list in the hierarchical list pane.

You can right-click a user in the List tab to display a shortcut menu where you can log the user out, or enable a user whose record is displayed.

3.8.4 Creating a User Group

Procedure

1. From the File menu, choose New, and then choose User Group to display a dialog box, where you can specify a unique name for the new user group.
2. Click OK Creating the user group.

The user group is added to the User Groups list in the hierarchical list pane — you can now add users to the group.

3.8.5 Deleting a User

Procedure

1. In the hierarchical list pane, expand the Security list.
2. In the Users list, select the user that you want to delete. Or, click the List tab, and select a user.
3. Do one of the following:
 - From the File menu, choose Delete.
 - Display the shortcut menu for the selected user, and then choose Delete.
 - Press the Delete key.
4. (Optional) To update the user details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

3.8.6 Removing a User from a User Group

If you want to remove multiple users from a group, use the User Group Properties dialog box.

Procedure to use the User Group Properties dialog box

1. In the hierarchical list pane, right-click the required user group to display a shortcut menu, and then choose Properties to display the User Group Properties dialog box.
2. Do one of the following:
 - To remove a single user from the user group, select the user in the right pane of the dialog box, and then click the Remove button.
 - Click the Remove All button to remove all the users from the user group.
3. Click OK to close the User Group Properties dialog box.

Procedure to remove a user from a user group

1. In the hierarchical list pane, expand the Security list, and then expand the User Groups list.
2. Make sure the List tab is displayed.
3. In the hierarchical list pane, click the name of the required user group to list the users in the group in the List tab.
4. In the List tab, right-click the required user to display a shortcut menu, and then choose Remove From Group.
5. If you have "Confirm before deleting an object" selected in the Options dialog box, you will be prompted to confirm the removal of the user from the group. Do one of the following:
 - Click Yes to remove the user from the group.
 - Click No to cancel the operation.

3.8.7 Removing a user group

Procedure

1. In the hierarchical list pane, expand the Security list.
2. In the User Groups list, select the user group that you want to delete.
3. Do one of the following:
 - From the File menu, choose Delete.
 - Display the shortcut menu for the user group, and then choose Delete.
 - Press the Delete key.
4. (Optional) To update the user group details in the List tab, click the List tab, and then click the Refresh button on the toolbar.

Each dimensional model has a default access mode. The current model name and access mode are shown on the status bar. The following access modes are available:

Read — Multiple users can read from the database, but cannot update it.

Shared — Multiple users can read from the database, and they can update report sets, time sets, documents, and procedures. However, users cannot update data or change the structure of the database by editing dimension sets.

Exclusive — One user can read from database, and update sets and data within it.

Once a database is opened for a particular level of access, all users must access the database at the same level. For example, if the first user accesses the database in exclusive mode, no other users can access the database. If the first user accesses the database in shared mode, other users can access the database in shared mode but they cannot be restricted to read access.

3.8.8 Changing your access mode

Procedure

1. In the Explorer window, expand the list of dimensional models.
2. Select the dimensional model you want to use.
3. Display the shortcut menu for the dimensional model, and then choose one of the following:
 - Use Read-Only. This is the default value.
 - Use Shared.
 - Use Exclusive.

Notes:

You can change the current access mode by issuing a USE command in the IDQL tab.

If you are logged in as Supervisor, you can change the default access mode for a dimensional model. For more information, see *Specifying the level of access for a database*.

3.8.9 Adding a Model to MASTERDB

Procedure

1. Right-click the hierarchical list pane to display a shortcut menu, and then choose Add Model to display the Add Model dialog box.

2. On the Model tab, specify the name of an existing model in the "Model name" text box to have Server search the DBPATH to find the model
3. On the Partitions tab, you can specify the model location — this is useful if the model has multiple partitions.

4 Application Server at the Command Level

4.1 Executing Commands in Application Server

There are two ways to execute Application Server commands: you can type a command directly in the IDQL tab or you can create a procedure using the Procedure editor.

Procedure to enter commands from Application Server

1. In Application Server, click the IDQL tab.
2. In the Command input window, type the Application Server command.
3. To execute the command, press Enter, or click the Execute button on the Application Server toolbar.

The Output window displays any output resulting from the IDQL command.

Notes:

You can use the command history to redisplay previously entered commands.

If you are running Application Server either in standalone mode or as client/server to a Windows NT computer, you can click the Stop button on the Application Server toolbar to stop an executing IDQL command.

Procedure to use the procedure editor

A procedure is a file that contains a list of valid Application Server commands with each command entered on a separate line.

4.2 Running Application Server While Executing DOS Batch Jobs

Procedure

1. Create and run procedures using the procedure editor.
2. Pass parameters into procedures to set the values of control variables.
3. Run Application Server in batch mode.

Note: Follow these steps to automate any Application Server process.

4.3 Building an Application Server Database

The analysis and design stage is an iterative process — you must involve end users to identify critical application reporting requirements.

Procedure

1. Identify the dimensions and variables in the database based on existing and planned reporting requirements.
 - a. Identify potential dimensions that reflect business reporting structures.

- Use existing reports columns and headings.
 - Determine whether you need to see X by Y.
 - Include a maximum of eight user-defined dimensions per variable.
- b. For each dimension, define the rollup hierarchies and other relationships.
 - Draw the hierarchies and identify levels by name.
 - Identify all the idiosyncrasies such as uneven hierarchies and classes.
 - Include a maximum of 20 levels in each dimension hierarchy.
 - c. Estimate the number of input and output members in each dimension.
 - d. Identify the data variables, which are the business measurements.
 - Determine how they relate to dimensions. For example, are you dealing with actual versus budget or shipped versus booked measurements?
 - Identify input and calculated variables.
 - Determine whether calculated variables need to be stored in the database.
 - Determine the calculation rules .Are there hidden variables needed for calculations?
 - Identify variable attributes - time conversion: first, last, and sum.
 - Determine if there is a hierarchical relationship between variables.
 - Include a maximum of 1,000 variables in a database.
2. Estimate the size of the database to see whether your data will fit in one database. Estimate consolidation speed to determine whether updating requirements can be met.

Note: The goal of this first round of design is to discover any limits which would prevent you from using one Application Server database.
 3. Examine any exceptions or idiosyncrasies that may affect the design. Handle issues such as having multiple databases for security, updating, or detail reporting requirements.
 4. Create the database.
 5. Create users who can access the database.
 6. Set the fiscal year.
 7. Create a dimension.
 8. Create variables.
 9. Load the data.
 10. Calculate and consolidate the variables.

4.4 Databases and Users

4.4.1 Analyzing Your Data

Before you can load data, you must identify the information that you have, and then identify the way you want it displayed in Application Server.

You should have all the dimensions and the data in one file so that you can read the file in just once. If you have information in various .DBF files, you must join them together from a common field using an appropriate program, such as Microsoft Excel. For example, the DRINKS.DBF file contains all the information necessary to build dimensions and variables.

By analyzing the data, you would determine that the business items of interest are PRODUCT, CHANNEL, and REGION; these are the dimensions. The DATE field will be used for the TIME dimension. The data in the ACTUNITS, ACTSALES, BUDUNITS, and BUDSALES columns are the business measurements; these are the variables. Based on the DRINKS.DBF file, you might determine the following reporting structure: the Product and Channel dimensions have simple reporting levels, the lowest level of data is consolidated into a total number, and the REGION dimension has several hierarchical levels.

Note: The field names that appear in the .DBF file must be the same as the dimension names you are constructing. For example, the .DBF file has a field called DATE, and Application Server has a dimension called TIME. You can either change the name of the DATE field in the .DBF file to TIME or you can use a CREATE command while you are accessing external data.

4.4.2 Creating a Database

Enter the CREATE DATABASE command.

The CREATE DATABASE command has many options that allow you to create an optimal database. The following command shows the options that make up a *default database*. In most cases, these settings should be increased to create a sufficient working database:

**CREATE DATABASE *database* BLOCKS 200 BLKSIZE 8k MEMBERS 10000 MULTIPLE 6
OBSERVATIONS 100 SIZE 32 USAGE Exclusive SORTKEYS Alphabetic VARIABLES 1000**

This default database will have the following characteristics:

BLOCKS 200	The database will have 200 blocks of physical disk space. Because Application Server does not preallocate blocks, and only uses them on an as needed basis, you should increase the BLOCKS number.
BLKSIZE 8k	Each block will be 8K. You should use this blocksize because it causes less I/O.
MEMBERS 10000	The largest dimension will have up to 10,000 members. You can specify an unlimited number of dimension members. You should modify this setting based on the number of members in your largest dimension.
MULTIPLE 6	Specifies the number of values that can be added to a time series before the record size is increased, where multiple is a numeric value between 1 and 255.
OBSERVATIONS 100	Specifies the maximum number of observations a time series can contain is 100. To determine this number, multiply the periodicity of each variable by its date range. Note: In most cases, these numbers are too small and impractical to build a sufficient database.
SIZE 32	The maximum size of the database can grow to 32 GB. You should set this parameter to the anticipated size of your database, not larger.
USAGE Exclusive	The user who created this database has exclusive read and write access to it. This mode is useful during the development phase. When you want others to use the database, you would change the USAGE to READ or SHARED.
SORTKEYS Alphabetic	The dimensions will be referenced internally in the database key structures in alphabetical order rather than based on the number of dimensions. You should specify SORTKEYS Nmembers rather than SORTKEYS Alphabetic to optimize your database.
VARIABLES 1000	You can create up to 1,000 variables in a database by default if you do not specify the VARIABLES keyword. Typically, this is more than sufficient. The maximum number of variables in a database is 10,000 variables.

4.4.3 Creating a Database with Multiple Partitions

By default, Application Server creates a database with a single partition. The partition is a disk file whose name is the same as that of the dimensional model, and is stored in the home directory.

When you specify a number after the PARTITION keyword, Application Server creates multiple partitions and names them <database>##, where <database> is the database name and ## is a two-digit suffix that is incremented by 1 for each partition. If you create 100 or more partitions, Application Server names them <database>####. Therefore, you must ensure that the name of the database contains six characters or less (five characters or less when used with 100 partitions or more).

In Windows, Application Server creates all partitions in the home directory. Once the partitions have been created, you can move them to other drives or directories. You must then create corresponding entries in the [Windows] section of LSSERVER.INI in the following format:

[Windows]

<database>## = pathname\<database>##

Substitute <database>## with the name of the partition you moved, and substitute pathname with the directory into which you moved the partition.

In UNIX client/server, you must define environment variables in lsstcp.sh.

Notes:

A database can have a total of 32 partitions.

You can display partition information with the SUPERVISOR SHOW DATABASE command.

4.4.4 Determining Buffers Setting

The amount of memory used by Application Server is determined by the number of buffers it creates in memory, and the size of the buffers. Application Server creates the maximum number of buffers possible in the memory available to it, up to the value set for buffers. To optimize data load and consolidate performance, you should set buffers as high as possible, but at a level that ensures they are created in real memory rather than cache or swap memory.

Available memory and buffers

The value you specify for the DEFAULTMEMORY environment variable or the SET MEMORY command determines the amount of memory available to Application Server. If the variable DEFAULTMEMORY is not set at startup, Application Server uses all available memory to create buffers.

If Application Server runs out of real memory for creating buffers, it uses cache or swap memory, which is less efficient and should be avoided. When determining the available memory value, you should reserve enough real memory for the operating system and priority processes, while ensuring that Application Server has as much real memory available to it as possible.

The maximum number of buffers that can be created in available memory is initially determined by the Buffers value on the Application Server user account. The default value of 2,000 is usually sufficient for query users. However, you can use the SET BUFFERS command to increase buffers for a specific process, such as a data load and consolidate. The SET BUFFERS command sets buffers for the current session only — the next time the user logs in, the Buffers value on the user account is used.

Determining buffer size

The size of the buffers that Application Server creates in memory is determined by the page size of the database set that stores time series records. The page size is determined by the observations value that is set when the database is created, and by the block size of the database. A page is made up of the number of database blocks taken to store the maximum number of observations for an 8-byte variable. The calculation of page size is based on 8 bytes, even if the database has only 4-byte variables.

For example, if observations is set to 2,000, then page size must be at least 16,000 bytes (that is, 8 bytes * 2000 observations). If the block size of the database is 8K, there are two blocks per page, and the buffer size is 16K. If observations is set to 1,500 and block size is 8K, page size and buffers are again 16K. Only 12,000 bytes (that is, 8 bytes * 1500 observations) are needed to store a fully populated time series, but page size will always be made up of full blocks. If block size is set to 4K, a database with 1,500 observations will have a page size and buffer size of 12K.

Determining the number of buffers

Once you know both the buffer size and the amount of memory available to Application Server, you can divide memory by buffer size to determine the approximate number of buffers to set for optimum performance. If you have set a value for DEFAULTMEMORY, or issued a SET MEMORY command, you should ensure that you set buffers high enough to take advantage of this setting. If you set buffers too high in relation to the amount of available memory (if buffers * page size is greater than the available memory), Application Server will not be able to create all the buffers you have requested.

To determine the observations setting for the model, use the Supervisor command SHOW DATA.

To determine the DEFAULTMEMORY or SET MEMORY value, and the Buffers setting for your Application Server session, use the SHOW SETTINGS command.

The importance of determining the correct buffer size

Although it is important to have sufficient memory to perform an operation, there can be a cost if you set your buffers too high.

When you issue the SET BUFFERS command, as well as allocating memory for buffers, Application Server allocates a table in contiguous memory with a 54-byte entry for every buffer. For example, if you set 2,000 buffers, the table contains 2,000 entries and occupies 108,000 bytes of memory.

Each allocated buffer that you do not actually need prevents 54 bytes of memory from being used by other applications. For example, if you set 3,000 buffers but need only 2,000 buffers, you allocate 54,000 bytes of memory to Application Server that it does not require and which cannot be used by other applications. For most computers, this relatively small amount of memory does not cause a problem, but it can become an issue if you set buffers to be much higher than actually required. Therefore, you should accurately determinate the buffer size your database actually requires.

4.4.5 Creating a User

Enter the CREATE USER command.

When you create a user in Application Server, a record is created and stored in the MASTERDB. This record contains information about the Login and the Use database. Since this record references a Use database, it must be created after creating the database. To create a new user, you need the following information:

The name of the Use database

The User ID to be assigned

The password for that user

The number of blocks to be allocated for the work database

The name for the work database

4.5 Dimensions

4.5.1 About Dimensions

Dimensions are components of a dimensional model. When building a dimensional model, create structural dimensions to describe an area of interest within your business, or attribute dimensions to describe a characteristic of a structural dimension.

Dimensions consist of members organized into a hierarchical structure. A dimension's hierarchical structure is the road map for drilling up and down on information.

4.5.2 Creating a Dimension Manually

Procedure

1. Enter the DIMENSION command followed by the dimension set name. The name should indicate the contents of the set and be a familiar term within your organization. A set name can have up to 96 bytes.
2. Enter the ALLOCATE command to define the maximum number of input, output, and result members in a dimension so that if you later add members to the dimension, the database is not reorganized. This statement must be the first line in the dimension and state the number of estimated input, output, and result members.
3. Enter the INPUT keyword followed by the member names separated by commas.
4. Assign alternative labels to the members of a dimension. These labels must be in single quotation marks (' '), and must follow the corresponding member name. When displaying data, you can use the SET SHORT or SET LONG commands to print either the member name or the label.
5. Enter the OUTPUT keyword before the output members and names. As with input members, output members must be separated by commas, and can have labels associated with them.
6. Define the RESULT <member>.
7. Define any LEVELS in a dimension. Level names should be descriptive and refer to the reporting structure. Names must also be unique within the dimension. The first name in the list corresponds to level 1, the second name to level 2, and so on. The result member or top level is not named in the statement.

4.5.3 Using a Procedure to Construct a Dimension

Create the following procedure, which accesses the external source, defines the structure of the dimensions, and then compiles the dimension:

ACCESS DBASE

USE drinks

CONSTRUCT Product LEVEL product

CONSTRUCT Region LEVEL city, stcode, region, country LABEL ?, state, ?, ?

CONSTRUCT Channel LEVEL channel

END

COMPILE DIMENSION Product

COMPILE DIMENSION Region

COMPILE DIMENSION Channel

This procedure is discussed in detail in the topic Constructing a dimension from a .DBF file, and an alternate procedure is provided in the topic Constructing a dimension with Link.

4.5.4 Building a Dimension Dynamically

Procedure

1. Enter the ACCESS EXTERNAL command.
2. Enter the USE command to identify the external file.
3. Enter the DESCRIPTION command to identify the layout.
4. Enter the CONSTRUCT command to identify the dimension name, levels, and labels.
5. Enter the END command to leave the Access window.
6. Enter the COMPILE DIMENSION <dimension name> command to permanently store the dimension.

4.5.5 Constructing a Dimension from a .DBF File

Procedure

1. Start the ACCESS subsystem:

ACCESS DBASE

2. Specify the file name without a .DBF extension. For example:

USE drinks

Note: The file must exist in a path specified by an environment variable that exists in the Application Server initialization file. For example, DRINKS.DBF might exist in the \Program Files\Pilot Software\Common directory.

3. Construct the first dimension in the file:

CONSTRUCT name LEVEL input, output, output....

For example, if the file looks like this:

CONSTRUCT Geog LEVEL city, state, region, country

The LABEL statement performs a one-to-one correlation with the LEVEL statement.

Notes:

- If no labels exist for the dimension, do not specify a LABEL statement. If labels exist for some but not all the levels in the dimension, use question marks to show that there is no label information for that level.
 - The preceding example shows that labels exist only for the state level. That is, the short name or member name is the stcode column. This column contains state abbreviations such as MA and FL. The labels are the fully spelled out words such as Massachusetts and Florida.
 - By default, a result member is created that sums the members of the last output level.
4. Close the ACCESS subsystem:
END
 5. Compile the dimension.

4.5.6 Constructing a Dimension with Link

Use ACCESS LSLINK to construct dimensions from a .DBF file, as shown in this procedure:

ACCESS LSLINK

CONNECT dbase

SELECT product, channel, country, region, stcode, state, city, date, actunits, actsales, budunits, budsales FROM drinks

PEEK ONLY 10

SELECT product, channel, country, region, stcode, state, city, date, actunits, actsales, budunits, budsales FROM drinks

TS CREATE time=date

PEEK CREATE ONLY 10

SELECT product, channel, country, region, stcode, state, city, date, actunits, actsales, budunits, budsales FROM drinks

CONSTRUCT product LEVEL product

SELECT product, channel, country, region, stcode, state, city, date, actunits, actsales, budunits, budsales FROM drinks

CONSTRUCT channel LEVEL channel

SELECT product, channel, country, region, stcode, state, city, date, actunits, actsales, budunits, budsales FROM drinks

CONSTRUCT region LEVEL city, stcode, region, country LABEL ?, state, ?, ?

END

COMPILE DIMENSION product

COMPILE DIMENSION channel

COMPILE DIMENSION region

4.5.7 Verifying a Dimension Structure

Procedure

1. Enter DIMENSION Product. The dimension appears in the editor.
2. Verify that the consolidations sum up the input members into output members:

INPUT

COLA
,UNCOLA
,SPRITZ
,DIET_DRINKS
,VALLEY_DEW

RESULT

TOTAL_PRODUCT 'TOTAL PRODUCT'

LEVEL

PRODUCT

TOTAL_PRODUCT = INPUTS

3. Choose Exit from the File menu.

4.5.8 Removing a Dimension

To remove a dimension's symbolic reference, but not the actual compiled version, enter:

REMOVE dimension dimension name

To force the removal of the actual compiled dimension set, enter:

REMOVE dimension dimension name sure

4.5.9 Saving a Dimension Set

Procedure

1. Choose Save from the Dimension menu. Another way to save the dimension structure is to choose Exit from the Dimension menu. Application Server confirms any changes.

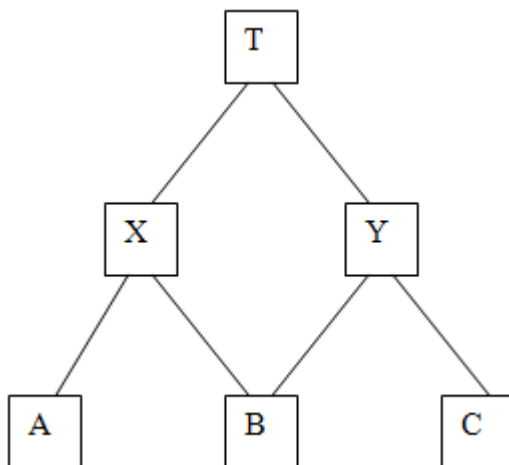
Note: In addition to Save and Exit, the Dimension menu has options that allow you to validate the dimension syntax, quit without saving changes, and enhance the readability of the set.

2. Choose EXIT from the dimension set to compile the set. A message stating the number of members selected appears. If you do not see that message, it means the set has not been compiled. To force the compilation, use the COMPILE dimension <set name> command.

4.5.10 Multiple Counting in Dimensions

In dimensions and hierarchical attributes, it is possible to have members that roll up into multiple parents. If any of those parents (or their ancestors) roll up into a common member then the hierarchy has multiple counting issues.

For example, suppose you have a dimension PRODUCT that has a hierarchy like this:



Input members A and B roll up into Output X and Input members B and C roll up into Output Y. X and Y roll up into a Result T. Suppose that a measure SALES is dimensioned by PRODUCT and that monthly sales of A, B and C are all 1 unit for November 2009.

If simple additive aggregation is performed then X is 2 and Y is 2, and then T is 4. However there were only 3 sales (A, B and C).

Since Input member B aggregates into Output X and Y, and Output X and Y aggregate into Result T, then sales of B contribute to T twice – once from X and once from Y. This hierarchy has double counting of B.

You have a choice over how you want dimensions counted when members roll up into multiple parents. By default, when a dimension is compiled and consolidated, the system reviews all multiple counting issues and counts just once. In terms of the example above, the Result is 3 by default.

If you want the system to include multiple counting even when there are multiple parents, you can use the NOCORRECTIONS keyword on the COMPILE DIMENSION command and CONSOLIDATE any measures that have dimensions with multiple counting issues in them. In terms of the example above, a COMPILE DIMENSION NOCORRECTIONS produces a Result of 4.

If you want to turn off automatic multiple counting in User-Defined Hierarchies then use the NOCORRECTIONS keyword on the CREATE command.

4.5.10.1 Displaying the automatic corrections

The SHOW CORRECTIONS command lists all members of a dimension that require automatic multiple counting corrections together with a list of each ancestor that requires adjustment and a correction factor.

4.5.10.2 Turning off automatic multiple counting corrections

If you actually want the hierarchy to include the multiple counting and PAS to behave as it did in prior versions then compile dimensions using the NOCORRECTIONS keyword and. PAS will not search for multiple counting issues in dimensions compiled with NOCORRECTIONS and all subsequent aggregations performed on those dimensions will be aggregated exactly as defined in the hierarchy with no corrections.

4.5.11 Identifying Which Dimensions Are in the Database

Enter the SHOW DIMENSION command.

4.5.12 Identifying How Dimension Members Are Defined

Do one of the following:

Enter the DIMENSION command followed by the dimension name. The dimension editor appears. At this point you see the dimension definitions or consolidation structure. To exit the dimension editor, choose Exit from the File menu.

Enter the TYPE command followed by the dimension name. For example:

TYPE (DIMENSION) MERCHANDISE

4.5.13 Identifying the Members of the Type Dimension

Enter the SHOW DIMENSION TYPE command.

The TYPE dimension contains two input members and one output member.

4.6 Attributes

4.6.1 About Attributes

Attributes provide a sophisticated and optimized database design for the following reasons:

They save disk space. Dimensions and variables require Application Server to store multiple combinations; this increases the size of the database and reduces disk space. Attributes have the same characteristics as dimensions and variables, but they do not store combinations; therefore, attributes reduce the size of the database. Attributes require significantly less storage than dimensions, multiple hierarchies in a dimension, dimension classes, and text variables.

You can solve business problems you could not otherwise solve. For example, when you look at your data, you might identify 20 items as dimensions. Generally, 20 dimensions are not possible or practical. Of the 20 items, the most significant in terms of characterizing the data can be defined as dimensions. The other items can be defined as attributes if they have a finite number of values which occur multiple times as data values. For example, PRODUCT would be defined as a dimension, but SIZE would be defined as an attribute, with many products sharing the same value for SIZE.

Attributes provide an easy conceptualization of certain business information. Attributes logically categorize information that describes features of a business entity.

Overview of working with attributes

You create an attribute by constructing it, compiling it, and then reading data into it. The process is similar to creating a dimension and reading data into a variable. You can construct attributes and read data into variables that store attribute information at any time once you have created dimensions. The following list provides general guidelines for the process:

1. Create attributes for a single source, multiple sources, or with hierarchies.
2. Check that the attribute was constructed properly.

3. Read data into the attribute variables.
4. List the attribute sets.
5. Set up a view containing the attributes.
6. Select the attributes.
7. Exhibit the attributes.

4.6.2 Source File Setup Considerations

If all the information about the attributes associated with a dimension exists in one source file, you can construct a single attribute set. If your attribute data is not in one file, you might want to join files together to benefit from this format. If it is not practical to combine all information into one file, you must construct one attribute set for each file.

Having all the attributes in one set poses a drawback when you update an attribute. To add or remove an attribute or an attribute member from the set, you must reconstruct the entire attribute set. If you expect your attributes to change, even if they are in one file, you might prefer to construct the attributes individually so that they exist in separate attribute sets. This way, if an attribute changes, you only have to reconstruct the individual attribute set.

4.6.3 Creating attributes

4.6.3.1 Creating Attributes for a Single Source

This topic assumes that:

You have already created the dimension with which the attribute will be associated. You have also created the variables associated with this dimension, read data into the variables, and consolidated the variables.

The attribute information exists in one file.

Procedure

1. Create a procedure. Enter the following:

```
PROCEDURE <setname>
```

where <setname> is the name of the procedure.

2. In the Procedure editor, specify the appropriate ACCESS subsystem and the source file containing attribute information:

ACCESS External	ACCESS LSLink
USE <filename>	CONNECT <i>name</i>
BEGIN	SELECT <field>,
DESCRIPTION	<field>, ...
<i>statements ...</i>	FROM
END	<filename>

3. Add the following line to construct the attribute:

```
CONSTRUCT ATTRIBUTE <setname> BY <dimension> VARIABLES <field>, ..., <field> LABEL  
    <field>, ..., <field>
```

where <setname> is the name of the attribute set. It can be the same name as the dimension it is associated with, or it can be any valid name you choose. For example, you might want to name the attribute set Prod_color to identify that the color attribute is associated with the Product dimension. If the set contains multiple attributes, you might name the set Prod_attr.

where <dimension> is the name of the dimension that this attribute will be associated with.

Attributes

VARIABLES <fields> are the column names from the source that contain the attribute information. You can specify multiple fields, separated by commas, to construct multiple attributes for the dimension. For example, VARIABLES color, size.

LABEL <fields> are optional. They are the source column names that contain attribute descriptions. You can specify multiple fields, separated by commas. This statement has a one-to-one relationship with the VARIABLES statement. That is, the first label field represents the label for the first attribute field. If some but not all attributes have descriptive columns, use a question mark (?) for the column that does not have a label.

4. Add a line to exit from the Access subsystem and compile the attribute set. Type:

```
END
COMPILE ATTRIBUTE <setname>
```

where <setname> is the name of the attribute you are constructing.

5. Save the changes to the procedure and exit from the Procedure editor by choosing File Exit.
6. Run the procedure. Type:

JOB setname

- Application Server creates an attribute set and one or more attribute variables. The attribute set identifies the members of one or more attribute dimensions and the name of the entity dimension the attributes are associated with. When you issue a single construct on a single source file, Application Server only creates one attribute set, which can contain information about any number of attribute dimensions. For example, you may create a single attribute set called PROD_ATTR that contains information about the attribute dimensions Color and Size.
- The attribute variables are created when you compile the attribute set. An attribute variable is a constant, 8-byte variable dimensioned by the entity dimension it characterizes. When you read data into it, it will contain the relationship between the members of the attribute dimension and the entity dimension. For example, the attribute variable Color stores the fact that Product P001 is red and P002 is blue. Attributes apply only to the input members of an entity dimension.
- The attribute dimensions and attribute variables share the same name, based on the field names you specified after the VARIABLE keyword in the CONSTRUCT command. If you specify multiple fields in the CONSTRUCT command, Application Server will create multiple attribute dimensions.

4.6.3.2 Creating Attributes from Multiple Sources

Procedure to create attributes when the source files have the same file type

If the attribute information for a dimension exists in multiple source files, you must construct separate attribute sets for each file.

Even if the attribute information for a dimension is in the same file, you might want to construct each attribute separately so that each one has its own attribute set.

Create a procedure that follows this format:

```
ACCESS <subsystem>
```

```
...statement(s) to select\describe the source
```

```
CONSTRUCT ATTRIBUTE <<setname1>> BY <dimension> VARIABLES <field>
```

```
...statement(s) to select\describe the next source
```

```
CONSTRUCT ATTRIBUTE <setname2> BY <dimension> VARIABLES <field>
```

```
END
```

COMPILE ATTRIBUTE <setname1>

COMPILE ATTRIBUTE <setname2>

Procedure to create attributes when the sources files have different file types

Create a procedure that follows this format:

ACCESS subsystem

...statement(s) to select\describe the source

CONSTRUCT ATTRIBUTE <setname1> BY <dimension> VARIABLES <field>

...statement(s) to select\describe the next source

END

COMPILE ATTRIBUTE <setname1>

ACCESS <subsystem>

...statement(s) to select\describe the next source

CONSTRUCT ATTRIBUTE <setname2> BY <dimension> VARIABLES <field>

END

COMPILE ATTRIBUTE <setname2>

Procedure to create individual attribute sets when all the attribute information exists in one source file

Create a procedure that follows this format:

ACCESS subsystem

...statement(s) to select\describe the source

CONSTRUCT ATTRIBUTE <setname1> BY <dimension> VARIABLES <field>

...repeat select statement if source is Access LsLink

CONSTRUCT ATTRIBUTE <setname2> BY <dimension> VARIABLES <field>

END

COMPILE ATTRIBUTE <setname1>

COMPILE ATTRIBUTE <setname2>

4.6.3.3 Creating an Attribute with Hierarchies

You can create an attribute with hierarchies. For example, an attribute called Flavor might have a base level containing flavors such as lemon and orange and a higher level containing flavor types such as fruit and non-fruit, and so on.

This topic assumes that:

You have already created the entity dimension that the attribute will be associated with. You have also created the variables associated with this dimension, read data into the variables, and consolidated the variables.

The attribute information exists in one file.

The process of adding hierarchical attributes to the database design involves the following steps:

1. Constructing the hierarchies of the attribute dimensions
2. Constructing the attribute sets
3. Reading data into the attribute variables

Procedure to construct an attribute with hierarchies

1. Create a procedure that constructs the hierarchies of the attribute into a dimension set:

```
ACCESS <subsystem>
'statement(s) to select/describe the source
CONSTRUCT <hierarchy_name1> LEVEL <field>, ..., <field> LABEL <field>, ..., <field> PREFACE
"BY <dimension>"
CONSTRUCT <hierarchy_name2> LEVEL <field>, ..., <field> LABEL <field>, ..., <field> PREFACE
"BY <dimension>"
END

COMPILE DIMENSION <hierarchy_name1>
COMPILE DIMENSION <hierarchy_name2>
```

where <hierarchy_name> is the name you want to call the hierarchical attribute. This name will be the name of the dimension set containing the attribute hierarchy members and level information. If you are constructing more than one hierarchical attribute for a dimension, specify each hierarchical attribute in a separate CONSTRUCT command.

where <fields> are the column names from the source that contains the hierarchy levels. You can specify multiple fields, separated by commas, to construct multiple levels for the attribute hierarchy. The first field is the lowest level of the hierarchy, the next field is the next level of the hierarchy, and so on.

PREFACE "BY <dimension>" is the name of the entity dimension that you are associating this hierarchical attribute with.

The result is a compiled dimension called <hierarchy_name> and an attribute variable with the same name.

Note: When you run the COMPILE DIMENSION command, the system checks for multiple counting issues in the hierarchical attribute dimension just like it does for standard dimensions. At run time, any dynamic aggregations performed will automatically adjust the aggregations to correct any multiple counting issues in a regular dimensions or hierarchical attribute dimensions in the view. For information, see [Multiple Counting in Dimensions](#).

The hierarchy dimension set looks like this:

```
BY <dimension>
INPUT
  '<member>'
  '<member> ...'
OUTPUT
  '<member>'
  '<member>'...
RESULT
  <TOTAL_hierarchy_name> '<TOTAL hierarchy_name>'
LEVEL
  <field>
  '<field>...'
  <consolidation statements>
```

2. Create a procedure that constructs the attribute set using the following format:

```
ACCESS <subsystem>
'statement(s) to select/describe the source
BEGIN
CONSTRUCT ATTRIBUTE <attributeset> BY <dimension> VARIABLE <field>, ..., <field> LABEL
<field>, ..., <field>
HIERARCHY <hierarchy_name>, ..., <hierarchy_name>
END
END
COMPILE ATTRIBUTE <attributeset>
```

where <attributeset> is the name of the attribute set you are constructing.

where <dimension> is the name of the entity dimension with which this attribute is associated.

VARIABLE <field> is the column name from the source that contains the attributes that do not have hierarchies. You can specify multiple fields, separated by commas, to construct multiple attributes for the dimension. For example:

VARIABLES color, size

HIERARCHY <hierarchy_name> is the hierarchical attribute name defined in the CONSTRUCT <hierarchy_name> command. If you constructed more than one hierarchical attribute, specify each name here, separated by a comma.

If you constructed a simple attribute and two hierarchical attributes, the attribute set looks like this:

```
BY <dimension>
VARIABLE <field>
  <member>
  ,<member> ...
VARIABLE <hierarchy_name> HIERARCHY
VARIABLE <hierarchy_name> HIERARCHY
```

4.6.3.4 Editing and Creating an Attribute Using the Set Editor

You can create an attribute using the set editor and you can edit a constructed attribute set at any time. For example, you can edit an attribute set to add a member to the list of possible values for the attribute.

Procedure

1. Enter the following command:

ATTRIBUTE <setname>

where <setname> is the name of the set you are creating or editing.

The Attribute editor appears.

2. Type or edit the following:

```
BY <dimension>
VARIABLE <field>
  <member> ['label ']
  ,<member> ['label'] ...
```

where <dimension> is the name of the entity dimension that the attribute is associated with.

where <field> is the name of the attribute.

where <member> is an attribute member. Specify one or more members separated by a comma.

where label is an optional label for the member. If your label contains single (' ') or double (" ") quotation marks, you need to encapsulate the member label in a different quoting character. For example: "Don't try this at home".

3. Save the changes and choose Exit from the File menu.

Application Server automatically compiles the attribute when you exit from the set editor.

4.6.3.5 Specifying a Class for an Attribute

Construct the attribute with the CLASS keyword. Use the BEGIN END construct to type the CONSTRUCT command on more than one line.

ACCESS <subsystem>

Attributes

...statements that select/describe the source

BEGIN

CONSTRUCT ATTRIBUTE <setname> **BY** <dimension> **VARIABLE** <field> **LABEL** <field> **CLASS** <field>

END

COMPILE ATTRIBUTE <setname>

where <field> is the column name of the class you are creating.

4.6.3.6 Checking that the Attribute Was Constructed Properly

When you construct an attribute, it creates an attribute set containing the constructed attribute dimensions. It also creates attribute variables for each dimension based on the fields you referenced from the source. You should check the attribute set, the attribute dimensions, and the attribute variables to make sure they were constructed properly.

Procedure to check the attribute dimensions

Enter:

EXHIBIT ADIMENSION

Procedure to check the attribute variables

Enter:

SHOW VARIABLES

Procedure to check the attribute set

1. Enter:

DIRECTORY ATTRIBUTE

Application Server displays the attribute sets. Optionally, you can enter **SHOW ATTRIBUTE**.

2. Go to the attribute set:

ATTRIBUTE <setname>

3. Check the structure. Multiple **VARIABLE** statements mean that this attribute set has multiple attribute dimensions and was constructed from one source file. If the attribute information was in separate source files, you will have separate attribute sets.

BY <dimension>

VARIABLE <field>

<member> '<label>'

, ...

,<member> '<label>'

VARIABLE <field>

,<member>, ...

CLASS <name> <member>, <member>, ...

where <dimension> is the name of the entity dimension this attribute is associated with.

where <field> is the name of the attribute. Application Server derives them from the source column name.

where <members> are the attribute members. Application Server derives them from the values in the source column.

where <labels> are the attribute member labels. They are optional. Application Server derives these labels from the values in the source column. If your label contains single (') or double (")

") quotation marks, you need to encapsulate the member label in a different quoting character. For example: "Don't try this at home".

4.6.3.7 Reading Data into Attribute Variables

After you construct an attribute set and the associated dimensions and variables, you can read data into the variables. The attribute variable stores the relationship between the attribute members and the associated entity dimension members. For example, the attribute variable color stores the value red for Product P001 and green for Product P002. You can load data into all attribute variables at the same time or into one variable at a time regardless of whether you constructed one or many attribute sets.

You read data into an attribute the same way that you read data into a variable.

Note: If attribute data does not exist for an entity dimension member, Application Server considers the data value missing.

Procedure

1. Create a procedure that reads data into the attribute:

```
PROCEDURE <setname>
SELECT VARIABLE <attributevar>, <attributevar>, ...
SELECT <dimension> INPUT
ACROSS VARIABLE DOWN Time, <dimension>
ACCESS <subsystem>
    <statements that select/describe the source>
PEEK ONLY 5
...Reselect the source, if necessary (Access LsLink)
    <statements that select/describe the source>
READ
END
```

where <attributevar> is the name of the attribute variable(s) whose data you are reading.

where <dimension> is the entity dimension this attribute is associated with.

Note: Because Time is a constant in this situation, it is the first dimension to appear in the Down list.

2. Save the changes and exit from the procedure editor by choosing Exit from the File menu.
3. Run the procedure:

```
JOB setname
```

4.6.3.8 Updating an Attribute

Procedure to add new attribute dimensions to an existing attribute set

1. Edit the procedure that constructs the attribute set.
2. Edit the CONSTRUCT ATTRIBUTE syntax and add a field to the list following the VARIABLES keyword.
3. Remove any other CONSTRUCT ATTRIBUTE commands that create separate sets and are not affected by the update.
4. Edit the procedure that reads data into the attribute variables and add the new attribute variable to the SELECT VAR list. Or, create a separate procedure to load the variable. Execute the new or modified procedure.

Procedure to add a new attribute to a new attribute set

1. Create a new procedure that constructs the attribute into its own attribute set.

2. Create and execute a procedure to read data into the attribute variable.

Procedure to remove one of many attributes in an attribute set

1. Edit the procedure that constructs the attribute set.
2. Edit the CONSTRUCT ATTRIBUTE syntax and remove the corresponding field from the list following the VARIABLES keyword.
3. Execute the modified procedure. Alternatively, you can manually edit the attribute set and remove the attribute by deleting the corresponding VARIABLE line and the list of members following it.
4. Remove the corresponding attribute variable with the REMOVE VAR command.

Procedure to remove an attribute that is in a separate attribute set

1. Remove the procedure that constructs the set. If the procedure constructs several attribute sets, edit the procedure and remove the corresponding CONSTRUCT ATTRIBUTE command.
2. Remove the attribute set with the REMOVE ATTRIBUTE command.
3. Remove the corresponding attribute variable with the REMOVE VAR command.

4.6.4 Using Attributes

4.6.4.1 Listing Attribute Sets

Enter either of the following commands:

DIRECTORY ATTRIBUTE

SHOW ATTRIBUTE

4.6.4.2 Setting Up a View that Includes Attributes

In the following examples, you see the difference between using Product dimension and the Color attribute in the DOWN list. The same SELECT commands are in effect in both examples.

These commands show the selected products as rows in the report:

SELECT sales

SELECT Product input

SELECT Product input

ACROSS time DOWN var, Product

ACROSS time DOWN var, Product

LIST

These commands show all the input members of product aggregated according to color. The sales of each color of products appear in separate rows in the output.

SELECT sales

SELECT Product input

SELECT Product input

ACROSS time DOWN var, Color

LIST

Note: You cannot display an attribute and a dimension with the same name in a view.

4.6.4.3 Selecting Attributes

Before you can display information about attributes, you must select the items for display and set up the across and down views.

Procedure

Type either of the following commands:

```
SELECT <dimension> WHERE <attribute> <expression> < member>, <member>
```

```
SELECT <attribute> <member>
```

where <dimension> is the name of the dimension you want to select.

where *attribute1* is the attribute you are using as criteria for selection.

where *attribute2* is the attribute you are selecting.

where *member* is the member of the attribute you are using as criteria for selection.

4.6.4.4 Exhibiting Attributes

Procedure to list the attribute set names in a database

Enter:

```
EXHIBIT ATTRIBUTES
```

Procedure to list the attributes and dimensions

Enter:

```
EXHIBIT DIMENSION
```

Procedure to list the entity dimensions

Enter:

```
EXHIBIT SDIMENSION
```

Procedure to list the attribute dimensions

Enter:

```
EXHIBIT ADIMENSION
```

Procedure to list the variables and attribute variables

Enter:

```
EXHIBIT VARIABLES
```

4.6.4.5 Removing Attributes

To remove an attribute set from a model enter the following:

To remove an attribute set that describes the attribute:

```
REMOVE ATTRIBUTE attribute_name
```

To remove the compiled attribute that results from compiling the attribute set:

```
REMOVE ADIMENSION attribute_name
```

To remove the attribute variable that is related to the compiled attribute and has the data that relates it back to the underlying structural dimension:

```
REMOVE VARIABLE attribute_variable_name
```

4.6.5 Examples for Attributes

4.6.5.1 Example: Constructing a Dimension with Hierarchical and Non-Hierarchical Attributes

The following sample procedure creates an attribute set for the Product dimension. The Product dimension will contain one non-hierarchical attribute named Brand, and three hierarchical attributes named Material, Color, and Style. The hierarchical attributes roll Products (the dimension) into various reporting structures:

Attribute Brand: Product	-> Brand	
SKU28592	-> Levis	
SKU11193	-> Calvin Klein	
SKU99100	-> Donna Karan	
Attribute Material:Product	-> Material	-> Material_Type
SKU28592	-> Denim	-> Cottons
SKU11193	-> Linen	-> Cottons
SKU99100	-> Raw Silk	-> Silks
Attribute Color: Product	-> Color	-> Color_Group
SKU28592	-> Charcoal	-> Darks
SKU11193	-> White	-> Lights
SKU99100	-> Beige	-> Lights
Attribute Style: Product	-> Style	-> Style_Group
SKU28592	-> Straight Cut	-> Classics
SKU11193	-> Loose Fit	-> Contemporary
SKU99100	-> Pleated	-> Classics

The sample procedure shown below is broken into three steps:

Create the Attribute Dimensions for the Hierarchical attributes

Create the Attribute Set, which includes the Hierarchical and non-Hierarchical attributes

Read data into the attributes

```

.....
... AUTHOR:  xxx
... DATE:   xxx
... PROC:   AC_PRODUCT
... DESC:   This procedure constructs the attributes Brand, Material,
...         Color, and Style for the dimension PRODUCT, then loads
...         data into the attributes.
...         All attributes are hierarchical except Brand.
.....
... Create a Trace to an external file for error processing

```

```

TRACE BOTH &TraceFile;ext overwrite
... Set up a TIMER in elapsed seconds
SET CONTROL TMR_AC_PRODUCT Clock
SHOW CONTROL TMR_AC_PRODUCT
.....
... STEP 1
... CONSTRUCT THE HIERARCHY FOR THE ATTRIBUTES
... This step creates the Attribute Dimensions required by
... Hierarchical Attributes.
... Only the Brand attribute is not hierarchical, so do not include
... Brand in this first step.
.....
... Starts the ACCESS Subsystem and Links to the correct Database
ACCESS LSLINK
CONNECT DataLink
... Issue the SQL select statement which selects fields
... from the Database table Products
BEGIN
    SELECT *
    FROM Product
END
...Build the Hierarchy for the Attribute Material
BEGIN
    CONSTRUCT Material
    LEVEL Material, Material_Type
    LABEL Material_Name, Material_Type_Name
    PREFACE "BY PRODUCT"
END
...Build the Hierarchy for the Attribute Color
BEGIN
    CONSTRUCT Color
    LEVEL Color, Color_Group
    LABEL Color_Name, Color_Group_Name
    PREFACE "BY PRODUCT"
END
...Build the Hierarchy for the Attribute Style
BEGIN
    CONSTRUCT Style

```

Attributes

```

    LEVEL Style, Style_Group
    LABEL Style_Name, Style_Group_Name
    PREFACE "BY PRODUCT"
END
... Ends out of the ACCESS Subsystem
END
...Compile the hierarchical attributes (not Brand)
COMPILE DIMENSION Material
COMPILE DIMENSION Color
COMPILE DIMENSION Style
.....
... STEP 2
... CONSTRUCT THE ATTRIBUTE SET
... This includes the Hierarchical and Non-hierarchical attributes.
.....
... Starts the ACCESS Subsystem and Links to the correct Database
ACCESS LSLINK
CONNECT DataLink
... Issue the SQL select statement which selects fields
... from the Product Database table
BEGIN
    SELECT *
    FROM Product
END
...Build the Attribute dimension set called PRODUCT
BEGIN
    CONSTRUCT ATTRIBUTE PRODUCT BY PRODUCT
    VARIABLE Brand
    LABEL Brand_Name
    HIERARCHY Material, Color, Style
    PREFACE 'RESULT LEVEL'
END
... Ends out of the ACCESS Subsystem
END
...Compile the attribute
COMPILE ATTRIBUTE PRODUCT
.....
... STEP 3

```

```

... READ DATA INTO THE ATTRIBUTES
.....
... Clear all prior selects
CLEAR STATUS
... Select dimension and variable for which data is to be loaded
SELECT PRODUCT INPUT
SELECT VARIABLES Brand, Material, Color, Style
... Attributes have no relation to time - they are constant over time
... Set the period to DEFAULT rather than selecting a date range
SET PERIOD DEFAULT
... Sets up external file matrix layout
ACROSS VARIABLES DOWN TIME, PRODUCT
... Starts the ACCESS Subsystem and Links to the
... correct Database
ACCESS LSLINK
CONNECT DataLink
... Issue the SQL select statements which selects fields
... from the Products Database table
BEGIN
    SELECT *
    FROM Product
    ORDER BY Product
END
... Reads the data into the Application Server database
READ SAVE 10000
... Ends out of the Access LSLINK subsystem
END
... Displays the TIMER in elapsed seconds
SET CONTROL TMR_AC_PRODUCT ELAPSED
SHOW CONTROL TMR_AC_PRODUCT
... Attribute PRODUCTS Complete
... Turns off trace to external file
TRACE BOTH OFF

```

4.6.5.2 Example: Constructing an Attribute with a Hierarchy

This example shows you how to construct dimensions, construct attributes with hierarchies, read data into variables, and read data into attribute variables.

Attributes

Consider a file that contains data for a Drinks and Region dimension. The file also contains data for an attribute called Pkg_Type and a hierarchical attribute that has a Size level that consolidates into a Package level. For example:

Drinks	City	State	Date	Size	Package	Pkg_Type	Sales
Cola	Phoenix	AZ	9801	32 oz.	Giant	bottle	1,000
Spritz	Boston	MA	9802	8 oz.	Junior	can	1,500

Column	Item to construct
Drinks	First level of the drinks dimension (Next level is total)
City	First level of the region dimension
State	Second level of the region dimension
Date	Time dimensions, automatically read into Application Server.
Size	First level in a hierarchical attribute
Package	Second level in a hierarchical attribute
Pkg_type	Another attribute of the drinks dimension
Sales	Sales variable

4.6.5.2.1 Constructing the Dimensions

Note: Use these procedures as syntax examples - they will not work with the demonstration databases in your installation.

This procedure constructs and compiles the Drinks and Region dimensions:

...Construct the Region dimension

ACCESS LSLINK

CONNECT dbase

SELECT city, state FROM Drinks

CONSTRUCT region LEVEL city, state

...Construct the Drinks dimension

SELECT drinks FROM Drinks

CONSTRUCT Drinks LEVEL Drinks

END

COMPILE DIMENSION Region

COMPILE DIMENSION Drinks

The following is an example of the dimension set for the Region dimension. Application Server consolidates two input members into a Result member.

INPUT

COLA

,SPRITZ

RESULT

TOTAL_DRINKS 'TOTAL DRINKS'

LEVEL

DRINKS

TOTAL_DRINKS = INPUTS

The following is an example of the dimension set for the Region dimension. Application Server consolidates two input members into an output level. Application Server consolidates the output level into a result.

```
INPUT
  PHOENIX
  ,BOSTON
OUTPUT
  AZ
  ,MA
RESULT
  TOTAL_REGION 'TOTAL REGION'
LEVEL
  CITY
  ,STATE
AZ = PHOENIX
MA = BOSTON
TOTAL_REGION = SUM AZ, MA
```

4.6.5.2.2 Constructing the Hierarchical Attribute

Note: Use this procedure as syntax example - it will not work with the demonstration databases in your installation.

This procedure constructs a dimension set containing the attribute member and level information for a hierarchical attribute. It also creates an attribute variable that bears the same name as the dimension set.

...Construct the attribute hierarchy associated with the Drinks dimension

```
ACCESS LSLINK
CONNECT dbase
SELECT drinks, size, package FROM Drinks
CONSTRUCT Size LEVEL size, package PREFACE "BY drinks"
END
COMPILE DIMENSION Size
```

The dimension set for the hierarchical attribute looks like this. It shows that sizes are the lowest level of the attribute's hierarchy, packaging is the next higher level, and a Result member is the highest level, which is the sum of all the members of the attribute.

DIMENSION Size

BY Drinks

```
INPUT
  '48_OZ.'
  , '32_OZ.'
  , '12_OZ.'
  , '8_OZ.'
OUTPUT
  'Junior'
  , 'Giant'
RESULT
```

Attributes

```

TOTAL_Size 'TOTAL Size'
LEVEL
  SIZE
  ,PACKAGE
'Junior'= SUM '8_OZ.','12_OZ.
'Giant'= SUM '32_OZ.','48_OZ.'
TOTAL_Size = SUM 'Giant', 'Junior'

```

4.6.5.2.3 Constructing the Attributes

Note: Use these procedures as syntax examples - they will not work with the demonstration databases in your installation.

This procedure constructs the attribute set, which contains a simple attribute and a hierarchical attribute. When the attribute set is compiled, the attribute variables pkg_type and size are also created.

```

ACCESS LSLINK
CONNECT dbase
SELECT Drinks, size, package, pkg_type FROM drinks
CONSTRUCT ATTRIBUTE attset BY drinks VARIABLE pkg_type HIERARCHY size
END
COMPILE ATTRIBUTE attset

```

The following is an example of the attribute set, attset. It shows that Application Server has constructed two attributes: the simple attribute Pkg_type and the hierarchical attribute Size.

```

ATTRIBUTE attset
BY DRINKS
VARIABLE PKG_TYPE
  BOTTLE
  ,CAN
VARIABLE SIZE HIERARCHY

```

This shows the output of a SHOW Size command:

```

SIZE
Inputs
  48 OZ. 32 OZ. 12 OZ. 8 OZ.
Outputs
  Junior      Giant
Result
TOTAL Size
  4 Inputs 2 Outputs 1 Result

```

This shows the results of a SHOW pkg_type command:

```

PKG_TYPE
Inputs
  BOTTLE CAN
  2 Inputs

```


4.6.5.2.4 Reading data into the Attribute Variables

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

...Select the attribute variables and then read in the data

```
SELECT var pkg_type, size
```

```
SELECT drinks input
```

```
ACROSS var DOWN Time, Drinks
```

```
ACCESS LSLINK
```

```
CONNECT dbase
```

```
SELECT * FROM Drinks
```

```
PEEK ONLY 5
```

```
LSS CREATE size = size
```

```
SELECT * FROM Drinks
```

...Read data into the variable

```
READ
```

```
END
```

...This shows the data that was loaded into the attribute variables:

```
PKG TYPE  HIER ATT
```

```
COLA  BOTTLE32_OZ.
```

```
SPRITZ CAN    8_OZ.
```

4.6.5.3 Example: Constructing an Attribute and Reading in data

This example discusses how to construct dimensions and attributes and read data into measure variables and attribute variables. The scenario uses three fictional .DBF files containing all the business information to load into an Application Server database. Users want to get reports showing which flavors of their products are selling the best and which region is selling the most of those popular flavors.

For the purposes of this example, this topic shows the information in each fictional .DBF file.

4.6.5.3.1 Studying the Source Files

Consider a file, PRODUCT.DBF, which contains information about the products that the company XYZ sells and attribute information about the products.

Column	Item to construct
Code	First level of the product dimension
Flavor	Attribute of the product dimension
Desc	Label for the Code column
Brand	Second level of the product dimension
Bdesc	Label for the Brand column
Pack_Cnt	Attribute of the product dimension
Form	Third level of the product dimension

Attributes

CODE	FLAVOR	DESC	BRAND	BDESC	PACK_CNT	FORM
950	ORANGE	ORANGE MINIS	22	CITRUS FRESH	6 count	BARS
953	ORANGE	ORANGE MINIS	22	CITRUS FRESH	5 pack	BARS
074	PEACH	SHER REG PK	44	Sherbetco	12 pack	Sandwich
075	RASPBERRY	SHER REG PK	44	Sherbetco	12 pack	Sandwich
076	STRAWBERRY	SHER REG PK	44	Sherbetco	12 pack	Sandwich
077	LIME	SHER REG PK	44	Sherbetco	12 pack	Sandwich
078	LEMON	SHER REG PK	44	Sherbetco	12 pack	Sandwich
079	LEMON	SHER REG PK	44	Sherbetco	12 pack	Sandwich
080	BLACKBERRY	SHER REG PK	44	Sherbetco	12 pack	Cups
081	ORANGE SWIRL	SHER REG PK	44	Sherbetco	12 pack	Cups

Consider a file, ACCOUNT.DBF, which contains information about the customers that the company XYZ sells to.

Column	Item to construct
Account_co	First level of the customer dimension
Name	Label for account_co column
Market	Second level of the customer dimension
District	Third level of the customer dimension
Region	Fourth level of the customer dimension
Zone	Fifth level of the customer dimension

ACCOUNT_CO	NAME	MARKET	DISTRICT	REGION	ZONE
4000	SHERBET	32608	BAY AREA	NORTHWEST	Western
8000	SELECT FD	32608	BAY AREA	NORTHWEST	Western
6000	N.E. DIST	12104	BOSTON	NORTHEAST	Eastern
6001	N.E. DIST	12104	BOSTON	NORTHEAST	Eastern
3000	FROSTY BITS	32608	BAY AREA	NORTHWEST	Western
3001	SEALED FDS	32608	BAY AREA	NORTHWEST	Western
3002	SEALED FDS	32804	S. CAL	WEST	Western
3003	SEALED FDS	32801	PHOENIX	WEST	Western
3004	SEALED FDS	32801	PHOENIX	WEST	Western
3005	SEALED FDS	32801	PHOENIX	WEST	Western
3006	SEALED FDS	32801	PHOENIX	WEST	Western

Consider a file, TRANSACT.DBF, which contains the sales data and broker commission data to read in.

Column	Item to construct
Code	Links the Code column in PRODUCT.DBF
Account_co	Links the Account_co column in Account.Dbf
Date	Time dimension automatically read in.

Sales Sales variable
 Broker_com Broker commission variable
 Cases Cases variable

CODE	ACCOUNT_CO	DATE	SALES	BROKER_COM	CASES	FORM
079	4000	9401	770.12	27.52	99.00	CUPS
081	8000	9401	540.24	35.04	198.00	CUPS
080	6000	9401	363.36	40.08	182.00	CUPS
076	6001	9401	333.76	80.94	91.00	BARS
077	3001	9401	887.52	62.82	273.00	BARS
075	3000	9401	908.00	204.00	200.00	DADE
078	3002	9401	804.00	42.00	100.00	DADE
950	3004	9401	908.00	94.00	200.00	DADE
074	3005	9401	908.00	94.00	200.00	DADE
953	3006	9401	976.00	77.80	160.00	SINGLES

4.6.5.3.2 Constructing the Dimensions

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

This procedure constructs the Product and Customer dimensions:

...Build the Product dimension

ACCESS LSLINK

CONNECT dbase

SELECT * FROM product

PEEK ONLY 10

SELECT * FROM product

CONSTRUCT product LEVEL code,brand, bdesc, NOALIAS

END

COMPILE DIMENSION product

...Build the customer dimension

ACCESS LSLINK

CONNECT dbase

SELECT * FROM account

PEEK ONLY 1

SELECT * FROM account

LSS CREATE cust = account_co

BEGIN

CONSTRUCT customer LEVEL cust, market, district, region, zone

LABELS name,?,?,?,?

END

END

Attributes

COMPILE DIMENSION customer

The dimension set for the Product dimension would look like this:

INPUT

'950' 'ORANGE MI'
, '953' 'ORANGE MI'
, '74' 'SHER REG'
, '75' 'SHER REG'
, '76' 'SHER REG'
, '77' 'SHER REG'
, '78' 'SHER REG'
, '79' 'SHER REG'
, '80' 'SHER REG'
, '81' 'SHER REG'

OUTPUT

'22' 'CITRUS FR'
, '44' 'Sherbetco'

RESULT

TOTAL_PRODUCT 'TOTAL PRODUCT'

LEVEL

CODE

, BRAND

'22' = SUM '950', '953'

'44' = SUM '74', '75', '76', '77', '78', '79', '80', '81'

TOTAL_PRODUCT = SUM '22', '44'

The dimension set for the Customer dimension would look like this:

INPUT

'4000' 'SHERBET'
, '8000' 'SELECT FD'
, '6000' 'N.E. DIST'
, '6001' 'N.E. DIST'
, '3000' 'FROSTY BI'
, '3001' 'SEALED FD'
, '3002' 'SEALED FD'
, '3003' 'SEALED FD'
, '3004' 'SEALED FD'
, '3005' 'SEALED FD'
, '3006' 'SEALED FD'

OUTPUT

'32608'
, '12104'

```
, '32804'
, '32801'
OUTPUT
  BAY_AREA
, BOSTON
, 'S._CAL'
, PHOENIX
OUTPUT
  NORTHWEST
, NORTHEAST
, WEST
OUTPUT
  WESTERN
, EASTERN
RESULT
  TOTAL_CUSTOMER 'TOTAL CUSTOMER'
LEVEL
  CUST
, MARKET
, DISTRICT
, REGION
, ZONE
'12104' = SUM '6000', '6001'
'32608' = SUM '4000', '8000', '3000', '3001'
'32801' = SUM '3003', '3004', '3005', '3006'
'32804' = '3002'
'S._CAL' = '32804'
BAY_AREA = '32608'
BOSTON = '12104'
PHOENIX = '32801'
NORTHEAST = BOSTON
NORTHWEST = BAY_AREA
WEST = SUM 'S._CAL', PHOENIX
EASTERN = NORTHEAST
WESTERN = SUM NORTHWEST, WEST
TOTAL_CUSTOMER = SUM WESTERN, EASTERN
```

4.6.5.3.3 Constructing the Attributes

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

Attributes

This procedure constructs two attributes, flavor and pack, stored in a single attribute set, prodatt, for the Product dimension:

...Construct the flavor and pack_cnt attributes

ACCESS LSLINK

CONNECT dbase

SELECT * FROM product

PEEK ONLY 10

SELECT * FROM product

CONSTRUCT ATTRIBUTE prodatt BY Product VARIABLE flavor, pack_cnt

END

COMPILE ATTRIBUTE prodatt

The attribute set looks like this:

BY PRODUCT

VARIABLE FLAVOR

BLACKBERRY

,LEMON

,LIME

,ORANGE

,ORANGE_SWIRL

,PEACH

,RASPBERRY

,STRAWBERRY

VARIABLE PACK_CNT

'12_PACK'

, '5_PACK'

, '6_COUNT'

The result of a SHOW VARIABLES command looks like this:

PRODUCT:

Variable	Type	Items	Date Range
FLAVOR	AT [8]	CO	
PACK_CNT	AT [3]	CO	

4.6.5.3.4 Creating the Variables, Reading Data into Them, and Consolidating

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

This procedure creates the sales, broker_com, and cases variables and reads data into the variables:

...Create the variables

CREATE sales, broker_com, cases BY product, customer

...Select the items and set up the view

```

ACROSS var DOWN product, customer, time
SELECT var BY product, customer
SELECT product input
SELECT customer input
SET PERIOD 1998
...Read data into the variable
ACCESS LSLINK
CONNECT dbase
SELECT * FROM transact
STATUS
PEEK ONLY 10
LSS CREATE product = code
LSS CREATE customer = account_co
LSS CREATE time = SUBSTR(date,1,2) + "/" + SUBSTR(date,3,2)
SELECT * FROM transact
READ ADD
END
...Consolidate
SELECT Product
SELECT Customer
SELECT var BY product, customer
ROLLUP sales
ADD EVERYBODY
END
CONSOLIDATE

```

The result of a SHOW VARIABLES command looks like this:

PRODUCT:

Variable	Type	Items	Date Range
FLAVOR	AT [8]	CO	
PACK_CNT	AT [3]	CO	

CUSTOMER, PRODUCT:

Variable	Type	Items	Date Range
BROKER_COM	NU	12 MO	Jan 98 - Dec 98
CASES	NU	12 MO	Jan 98 - Dec 98
SALES	NU	12 MO	Jan 98 - Dec 98

4.6.5.3.5 Reading Data into the Attribute Variables

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

This procedure reads information into the flavor and pack_cnt attribute variables:

```
...Read data into the flavor and pack_cnt attribute variables
```

Attributes

```
SELECT var flavor, pack_cnt
SELECT product input
SET PERIOD DEFAULT ACROSS var DOWN time, product
ACCESS LSLINK
CONNECT dbase
SELECT * FROM product
PEEK ONLY 10
LSS CREATE product=code
SELECT * FROM product
READ
END
```

4.6.5.3.6 Using Attributes

Assuming that the Product and Customer dimensions and the Pack_cnt and Flavor attributes are constructed in this fictional example, this section displays some possible sample report code:

Note: Use these procedures as syntax examples - they will not work with the demonstration databases in your installation.

What is the best selling flavor of sherbet this year?

```
SELECT sales
SELECT flavor orange
SELECT Customer region
ACROSS var, time DOWN flavor, customer
ORDER customer ON sales
LIST
```

Which region is selling the most orange sherbet this year?

```
SELECT sales
SELECT flavor orange
SELECT Customer region
ORDER Region ON sales
LIST
```

Note: You cannot display data for an attribute on a rate variable. For example, the following will not list the rate variable MGNPCT2:

```
USE JUICE
SELECT MON
SELECT PER JAN96
SELECT CHANNEL
SELECT CUSTOMER
SELECT PRODUCT
SELECT COT
CREATE MON MGNPCT2 BY CHANNEL, CUSTOMER, PRODUCT RATE
CALCULATE MGNPCT2=100*(sales-quota)/quota FULL
```



```

9140 Member Combination(s) Calculated; 370 Consolidated
SELECT CHANNEL #1
SELECT PRODUCT #1
SELECT CUSTOMER #1
ACROSS TIME,CHANNEL,CUSTOMER,PRODUCT,VAR DOWN COT
SELECT SALES, QUOTA, MGNPCT2
    1 Variable Currently Selected
LIST

        Jan 99
        Direct
        7 Eleven - Nashua NH
        Courtyard 12oz Conc.
        Sales      Quota      MGNPCT2
CONVENIENCE      11,524.20    617,904.00    -
TOTAL COT        11,524.20    617,904.00

```

4.7 Variables and Reading in Data

4.7.1 About Variables

Application Server uses three types of variables:

Normal variables — These are often referred to as derived variables because their data values are derived from relational tables. Data values are stored within time series in the database.

Calculated variables — These are calculated by Application Server by applying user specified formulas or logics to other variables in the database. For example, you might create a calculated variable named Margin, which is calculated using the formula Sales - Costs. Like normal variables, data values for calculated variables are stored within time series in the database. These data values are generated when you execute a calculate command.

Virtual variables — Like calculated variables, virtual variables are created by applying formulas and logics to existing variables. They differ from calculated variables because their data values are generated dynamically when you request them, and they are not stored permanently. For example, you might create a virtual variable named Margin% which is calculated using the formula $((\text{Sales} - \text{Costs}) / \text{Sales}) * 100$. Data values are automatically generated when you create a view that contains Margin%.

Distributed variables — From version 6.2 onwards, you have the ability to create distributed variables. These variables are contained within different databases. Data is distributed across multiple Application Server models. The master model, together with the slave (also called variable) models, make up a virtual model or cube.

4.7.2 Creating a Variable

Procedure

1. Open a dimensional model for exclusive access.
2. Enter the following command:

```
CREATE periodicity [TEXT] name BY dimensions
```

where:

periodicity is the frequency of data observations, such as monthly

TEXT is an optional keyword indicating the variable contains text

name is the name of the variable you want to create

dimensions is a list of dimensions, separated by commas

For example:

CREATE VARIABLE SALES WEEKLY BY MERCHANDISE, TYPE

Note: Application Server creates a default monthly numeric variable if you do not assign a frequency.

4.7.3 Creating a Virtual Variable

Procedure

1. Open the database for exclusive access.
2. Enter either of the following commands:

CREATE variable BY dimensions as expression

CREATE variable BY dimensions LOGIC logic

where

- variable is the name of the virtual variable you want to create
- dimensions is a list of dimensions the virtual variable is dimensioned by; separate dimension names with a comma
- expression is the expression you want to use to calculate the variable
- logic is the logic you want to use to calculate the variable

Notes:

You cannot specify expressions that are associated with time, or specify a long name for the virtual variable in quotes. Use a logic if you need to create a long name for the virtual variable.

Virtual measures do not support periodicity-based conditions, for example, WHEN YEAR IS XXX.

4.7.4 Using Virtual Variables

You create virtual variables by applying formulas and logics to existing variables. For example, you might create a virtual variable named Margin% which is calculated using the formula $((\text{Sales} - \text{Costs}) / \text{Sales}) * 100$. Virtual variables differ from calculated variables because their data values are generated dynamically when you request them. Because no data values are stored in the database, the database size is substantially reduced and its structure is simplified.

With normal variables, Application Server only provides support for summing attributes or User-Defined Hierarchies. Using virtual variables, you can create percentage and ratio variables (which do not use summation) and use them with attributes or User-Defined Hierarchies.

Notes:

Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.

Virtual measures do not support periodicity-based conditions, for example, WHEN YEAR IS XXX.

4.7.5 Using Distributed Variables

You can build an Application Server master model whose variables are contained within different databases — these are known as distributed variables. Variable data is distributed across multiple Application Server models. The master model, together with the slave (also called variable) models, make up a virtual model or cube.

Using distributed variables enables you to carry out parallel maintenance of the databases that make up the Application Server model. If you run multiple jobs in parallel on a multiprocessor machine, the time taken to load and consolidate data can be greatly reduced. This facility is not possible when the variables all exist in the same database.

Another benefit of using distributed variables is that the time series for each variable will be "clustered" together in their own databases, providing faster data retrieval by minimizing the number of database reads required to provide the typical types of views.

When building virtual models with distributed variables, you must always have both the master model and the appropriate slave, or variable, models attached appropriately. The master model is the model in which you construct and store dimensions, both structural and attribute, and virtual measures. A slave, or variable, model is the model in which you store variables.

Because each user's requirements will be different, a high level overview of the processes required to build a virtual model is provided as a reference for the Database Administrator (DBA).

Constraints in using distributed variables

When using distributed variables, you cannot change the name of the master or slave databases, as the master database stores the names of the slave databases, and vice versa.

Names of dimensions in distributed databases are limited to a total length of 24 bytes, which must include the name of the master database and a semicolon. For example:

product; mymaindb

Note: In the Japanese version of Application Server, the total name length is limited to 12 double-byte characters, including the master database name and semicolon.

You must issue separate ROLLUP commands for every database that contains variables. If your master model does not contain any variables but you want to ensure that the Rollup editor is in effect if they are created at any point, you can issue the ROLLUP command and pass it a dimension list in place of a variable name.

Commands to support the use of distributed variables

The following commands have additional keywords to support the use of distributed variables:

CHANGE DATABASE
CREATE DATABASE
CREATE VARIABLE
EXHIBIT DATABASES
EXHIBIT MEASURES
ROLLUP
SHOW VARIABLES
USE

4.7.6 Reading in Data

Procedure

1. Select the dimensions by which the variable data will be read in. For example:
SELECT Product
SELECT Region
SELECT Channel
2. (Optional) Enter the SHOW VARIABLE command to see a list of all variables in the database.
3. Select the variables into which data will be read. For example:
SELECT VARIABLES or
SELECT Actsales
SELECT Budsales
SELECT Actunits
SELECT Budunits
4. Specify the across/down structure from the source file. Variables are the across dimensions and time is the down. For example:
ACROSS var DOWN product, region, channel, time
5. Specify the period you want to read data in:
SET PERIOD 1/1/97 - 12/1/98
6. Start the Access subsystem:
ACCESS DBASE
7. Specify the source file to use:
USE drinks
CREATE TIME=DATE
8. (Optional) Make sure the first few records are being read in properly:
PEEK only 10
9. Read in the data:
READ
10. Exit from the Access Subsystem:
END

4.7.7 Reading an External File

Procedure

1. Match the layout of the Application Server selections to the layout of the source file using the SELECT and ACROSS/DOWN commands.
2. Set the associated data range with the SET PERIOD command.
3. Enter the ACCESS EXTERNAL command.
4. Enter the USE command to identify the external file.
5. Enter the DESCRIPTION commands to identify the source file field names, types, widths, and positions.

6. Enter the PEEK command to verify that the layout is correct.
7. Enter the READ command.
8. Enter END to return to the COMMAND window.

4.7.8 Loading the Data

Procedure

1. Match the layout of your Application Server selections to the layout of the record/field format of the external file.
2. Enter the ACCESS command for the appropriate data source.
3. Enter the USE command to identify the external file that you want to read in or write out to.
4. Describe the field layout with either the DESCRIPTION command or the CONNECT and SELECT commands.
5. Enter the PEEK command to look at the external file with the description applied to the file.
6. Enter the READ command to load the data into the database.
7. Enter the END command to return to the command window.

Procedure to load the data in variables

Use the ACCESS subsystem to load data stored in variables.

1. Select the variables and dimension members that must be selected.
2. Establish an appropriate data view.
3. Enter the READ command to update the selected time series.

Notes

You can increase the speed of loading data significantly by:

Pre-sorting the data file

Defining the best possible Across/Down layout

Matching the Down command to the sort order

Extending the work database size temporarily in the load procedure

Increasing the buffer size

4.7.9 Tracking Load Processing Times

Create a procedure to help you track load processing statistics.

Use the CLOCK and ELAPSED functions with control variables to track time statistics.

Use the TRACE command to track commands and reassurance messages to an external file that you can review when the job is complete.

4.7.10 Performance Considerations when Creating Variables

INTEGRAL BYTES 1 stores data as integers between 0 and 254. Use for flags or logic sets.

INTEGRAL BYTES 2 stores data as integers between 0 and 32,766.

INTEGRAL BYTES 4 stores data as integers between 0 and 2,147,483,647.

BYTES 4 stores data as floating point with 7 decimal digit precision.

BYTES 8 stores data as floating point with 15 decimal digit precision. This uses twice as much storage space as BYTES 4.

4.7.11 Identifying the Variables in the Database

Enter the SHOW VARIABLES command.

The SHOW VARIABLES command displays information regarding the attributes of each variable in the user's database. Variables are grouped together if they share the same dimension(s).

For each variable the following information is presented:

The variable name

The type (numeric or text)

Any necessary attributes

The number of observations in the time series

The periodicity and the date range associated with the data stored

Note: If a variable description has been entered or the variable is weighted by another variable, a description is also presented below the variable name.

4.7.12 Excluding Certain Variables in an EXHIBIT Command

Procedure

1. In Application Server, create a document called EXCLUDEVAR that contains a list of variables, virtual variables, or measures that you want to exclude when exhibiting variables or virtual variables. For example:

DOCUMENT Excludevar

Costs

Margin

2. From the File menu, choose Save, and then choose Close.
3. Issue an EXHIBIT MEASURES, EXHIBIT VARIABLES, or EXHIBIT VVARIABLES command. For example:

EXHIBIT MEASURES EXCLUDE – lists all virtual variables in the database, and all variables not included in the Excludevar document.

EXHIBIT MEASURES DATABASE myremotedb EXCLUDE – lists all virtual variables contained within distributed remote databases while excluding any variables listed in the Excludevar document.

EXHIBIT VARIABLES EXCLUDE - lists all variables and attributes in the database, and excludes any variables listed in the Excludevar document.

EXHIBIT VARIABLES LIKE SALES EXCLUDE - lists all variables and attributes in the database, and excludes any variables listed in the Excludevar document, if one exists

EXHIBIT VVARIABLES EXCLUDE - displays a list of all virtual variable properties, and excludes any virtual variables listed in the Excludevar document.

Note: You might want to exclude variables from the results of an EXHIBIT command if certain variables are used only in the calculation of other variables.

4.8 Data Calculation, Consolidation and Rollup

4.8.1 Calculating and Consolidating the Variables

The CALCULATE command calculates and consolidates all the variables in the equation. When you enter the CALCULATE command, two things occur:

The specific equation is calculated for every input dimension member selected.

Each variable defined in the equation is then automatically consolidated, if output, or result members, or both are selected.

Calculate Full Command

The CALCULATE FULL keyword forces the recalculation of output time series for the calculated variable. Whenever you issue the CALCULATE FULL command, three things occur:

The specific equation is calculated for every input dimension member selected.

Each variable defined in the equation is then consolidated, providing output, or result members, or both for the dimensions that have been selected.

The formula is recalculated at all consolidated levels of the dimensions that have been selected.

Consolidation

Application Server will only consolidate based on which variables and members have been selected for the specified time period. For example:

```
>SELECT SALES  
>SELECT TYPE  
>SELECT MERCHANDISE  
>CONSOLIDATE (SALES) PERIOD August 1995
```

If the dimension(s) of a variable has output or result members, the variable can have time series that store the consolidated values for the time-series created by output members.

The values for these consolidated time series are generated by the CONSOLIDATE command. Consolidation uses the data from the input member time series, and calculates the consolidated member time series according to the rules defined in the dimension set. Application Server consolidates selected variables. You can specify the variables to consolidate by including their name(s) with the CONSOLIDATE command. In addition, all time periods are consolidated unless you limit processing with the PERIOD keyword or SET PERIOD date range command.

4.8.2 Specifying a Fast Consolidation

You use the ADD statement in the Rollup editor to specify which member combinations (quadrants) to use in a fast (smart) consolidation. A smart or fast consolidation gets each series from the database for a particular variable and adds it to its parent.

Fast consolidation uses a strictly additive approach, where only simple additions and subtractions are used to roll up child series into their parents.

For example, a fast consolidation can execute statements such as:

CA = SUM LOS_ANGELES,SAN_FRANCISCO,SACRAMENTO,SAN_DIEGO

During the CONSOLIDATE, Application Server gets each time series available and adds it to the parent, in this case, CA. If data is not present for a particular combination, no work is required. For example, if data exists only for SAN_DIEGO, only one I/O operation has to be carried out to get the series. (A normal consolidation would require four I/O operations.)

The fraction of work saved is directly proportional to the sparsity of the data. For example, if only 1/10th of the possible series is available, Application Server spends only 1/10th of the effort.

Specifying a fast consolidation that consolidates all combinations

In the Rollup editor, issue the ADD statement. For example:

ROLLUP Sales

```
ADD EVERYBODY
CHANGE 2 PRECONSOLIDATED
END
```

CONSOLIDATE Sales

If all data needs to be consolidated, you would want to specify ADD EVERYBODY to include all member combinations. You would add only certain combinations if you know that you never need to look at consolidated data for a particular combination.

If all combinations are currently added and consolidated, and want to remove an unpopular or unnecessary combination to optimize the database, you can remove the combination from consolidation using the REMOVE statement. Or you can designate that the unpopular combination will be available for consolidation on the fly using the CHANGE DYNAMIC statement.

Note: You cannot use the Rollup editor to define rollups on variables where dimensions contain non-additive logic constructs (for example, WHEN...ENDWHEN, multiplication, and so on). Dimensions with different rules, such as multiplication, division, WHEN/ENDWHEN logic, complicated calculations, or logic constructs, are rejected.

4.8.3 Specifying a Normal Consolidation

A normal consolidation executes the consolidation statements defined in the dimension. You might want to perform a normal consolidation when data is loaded at the output level for certain combinations.

Procedure to perform a normal consolidation on data that has been consolidated by the fast consolidation method

1. Remove the dimension/quadrant table using:

```
ROLLUP <variable>
REMOVE EVERYBODY
END
```

2. Issue the CONSOLIDATE command.

Procedure to perform a normal consolidation on data that has not been consolidated yet

Issue the CONSOLIDATE command.

For example, a normal consolidation can execute the following statement in a Dimension set:

CA = SUM LOS_ANGELES,SAN_FRANCISCO,SACRAMENTO,SAN_DIEGO

During the CONSOLIDATE, Application Server by getting the time series for LOS_ANGELES, and then for SAN_FRANCISCO, and then for SACRAMENTO, and then for SAN_DIEGO. Then it adds the series and stores them in CA. A normal consolidation would require four I/O operations. It is more time consuming than a fast consolidation when data is not present for a particular combination because an I/O operation occurs for this combination anyway.

4.8.4 Specifying on the fly Consolidation

You can specify that certain member combinations should be consolidated on the fly. These combinations are not consolidated during a normal or fast/smart consolidation. Instead, they are consolidated whenever any commands requiring data about the combinations are issued. For example, a combination marked as dynamic (consolidated on the fly) would be consolidated on the fly for a LIST, DISPLAY, or CALCULATE command.

The aggregations are not saved in the database from session to session. This means that if you use the CALCULATE command for a combination that is dynamic, the calculated data will not be stored in the database either.

Use the CHANGE <number>s DYNAMIC syntax to specify that certain quadrants will be consolidated on the fly rather than consolidated during a normal or fast consolidation.

If you have a large database, and it typically takes a very long time to load data and consolidate the data, you should consider where it might be appropriate to consolidate certain combinations on the fly.

You might want to consolidate combinations on the fly in the following situations:

If you have a small dimension, and you want to keep the database size at a minimum. Because smaller dimensions have fewer series to consolidate, the process of consolidating that combination on the fly on an as needed basis would be quick.

When a particular combination, regardless of size, is not used frequently. This also minimizes the size of the database. The time it would take to consolidate on the fly a combination that is rarely used would be negligible. For example, if sales reps view the totals for sales in all regions every day, you would want to perform a fast consolidation on those combinations so that this information is always available as soon as possible. But if the sales in the South are rarely viewed, you can specify that combination to be on the fly.

You are updating or adding new data frequently, and do not wish to store lots of preconsolidated data which would require reconsolidating when the input data changes. If more of the data is consolidated on the fly, you will have to reconsolidate less data in your fast consolidations. This will be quicker, allowing you to load your new data, reconsolidate and have up-to-date consolidations available in a much shorter time.

Note: Virtual variables and attributes are also consolidated on the fly, by design.

Procedure to consolidate combinations on the fly

In the Rollup editor, issue the CHANGE DYNAMIC statement to specify the combinations that you want to mark as "on the fly".

In this example, the output level of the Channel dimension will be consolidated on the fly during a LIST command.

ROLLUP <variable>

CHANGE Channel Output DYNAMIC

END

ACROSS Tim, Var DOWN Product, Channel

LIST

When you issue a SHOW statement, the quadrants marked as on the fly have an exclamation point (!) next to them.

Procedure to change a combination's consolidation from on the fly to fast consolidation

In the Rollup editor, issue the CHANGE STATIC statement to change the combination's consolidation method from on the fly to a fast consolidation.

ROLLUP <variable>

CHANGE Channel Output STATIC

END

Note: After changing a combination back to static, you should perform a fast consolidate to preconsolidate that combination. Otherwise, subsequent LISTs or DISPLAYs or a DataView will not display any data for the combination.

4.8.5 Displaying Quadrant Information

When you enter ROLLUP <variable>, Application Server creates a table based on the dimensions associated with that variable and all other variables with the same dimensions. Use the SHOW statement to view the table. When you issue a ROLLUP statement in the Rollup editor, and then issue a SHOW, the table is updated with symbols to identify how the quadrants will be consolidated when a CONSOLIDATE command is issued.

Displaying quadrants, rollup instructions, and their percentage that they are already consolidated

In the Rollup editor, use the SHOW statement or SHOW COUNT.

Rollups> show count

#	CHANNEL	CUSTOMER	PRODUCT	%	# (9022)
1	Input	Input	Input	29.0	2616
2	Output	Input	Input	25.0	2256
3	Input	Output	Input	10.0	900
4	Input	Input	Output	13.4	1209
5	Output	Output	Input	5.2	468
6	Output	Input	Output	12.0	1079
7	Input	Output	Output	3.6	325
8	Output	Output	Output	1.9	169

The SHOW keyword displays the following symbols to denote the various types:

- * means that there is data entered for these input combinations.
- + identifies the dimension to be consolidated.
- & means that there's data input at output levels, but existing series will not be overwritten during consolidation. This identifies that a NOOVERWRITE keyword was issue.
- ! means that the quadrant is marked with the DYNAMIC keyword and will be consolidated on the fly.
- \$ means that the quadrant is marked as PRECONSOLIDATED so it will not be consolidated during a fast consolidation.

4.8.6 Removing Unnecessary Combinations from the Rollup

You can remove time-series combinations that Application Server does not need for reporting and analysis. By specifying the unnecessary combinations that you do not want to be consolidated, it reduces the size of the database and increases calculation speeds.

When you exclude the combinations for a specific variable, the combinations for all other variables with the same dimension structure are excluded.

If you want to perform a normal consolidation but some combinations are already consolidated, you need to remove all member combinations.

Procedure to remove all combinations

In the Rollup editor, issue the REMOVE statement. For example:

```
ROLLUP <variable>
REMOVE EVERYBODY or REMOVE 2,5,7
END
```

4.8.7 Specifying that Data Is Loaded at the Output Level

Typically, a dimension's input level contains the loaded, raw data. During a consolidation, the input level quadrants are consolidated into the output levels.

You can specify that particular quadrants have input data at the output level and that those quadrants will be consolidated from the output level. For example, you might have channel sales transactions without specific information about direct sales or distributor sales.

Specifying that a quadrant has data loaded at the output level for the purposes of consolidation

In the Rollup editor, issue the CHANGE UPDATE CONSOLIDATE statement to specify the level at which data is loaded for the consolidation. For example:

```
ROLLUP <variable>
CHANGE 3 UPDATE CONSOLIDATE Region Output
END
```

4.8.8 Consolidating Data

Procedure

1. Specify a CHECKPOINT command that freezes the current status of the database:
CHECKPOINT FREEZE CONTINUE
2. Select the dimensions and variables you want to consolidate. For example:
SELECT Goods
SELECT Zone
SELECT Actsales
3. Specify the consolidation command for the period of time to be consolidated. For example:
CONSOLIDATE Actsales PERIOD 1/1/99 - 12/1/00

4. Specify a CHECKPOINT command that updates the database:

CHECKPOINT UPDATE

5. Select other variables and continue consolidating. For example:

SELECT Goods

SELECT Zone

SELECT Budsales

CONSOLIDATE Budsales PERIOD 1/1/99 - 12/1/00

Note: The SET VARIABLE [CONSOLIDATE | NOCONSOLIDATE] and CHECKPOINT commands control the efficiency of a consolidation.

4.9 Date Ranges and Fiscal Year Settings

4.9.1 Setting the Fiscal Year

Enter the SET FISCAL command that describes your company's fiscal year setup.

If no fiscal year is set, the default is a standard January calendar year, and all time conversion calculations are based on this standard.

4.9.2 Identifying Your Fiscal Year

If a fiscal year other than the default (a calendar year starting in January) has been established, you can verify the fiscal year by entering the STATUS command. This command displays information about the fiscal year as well as about which databases are attached.

4.9.3 Combining a Period Range and Periodicity

Enter the LIST DAILY command to convert the variables selected to a daily periodicity.

For example, the following command would display the selected variables with a daily periodicity:

LIST daily

4.9.4 Changing the Periodicity of a Report

Do one of the following:

Enter the LIST command with a periodicity keyword.

For example, LIST MONTHLY lists the data monthly and returns to the default periodicity after the command is complete.

Use the SET command with a periodicity keyword.

For example, SET MONTHLY lists the data monthly. Any subsequent LIST commands display this new periodicity.

4.9.5 Resetting the Period Range

Do one of the following:

Enter the SET PERIOD DEFAULT command. The date range returns to the default setting.

Enter the CLEAR STATUS command. The date range is reset to your original login state.

4.9.6 Specifying a Date Range

Do one of the following:

Enter the LIST command with the PERIOD keyword.

Enter the SET PERIOD command to limit the displayed time frame.

Date formats include:

```
LIST PERIOD 00/1-00/6
```

```
SET PERIOD Jan 00 - Jun 00
```

```
SET PERIOD DEFAULT
```

```
LIST MONTHLY PERIOD 2000
```

Note: Application Server interprets most date conventions, but if the interpretation is not obvious, Application Server interprets dates in a MDY convention. Thus, if a date is entered as 1/2/00, it is interpreted as January 2 rather than February 1. You can enter the SET DATE command, followed by MDY or DMY, to change this default.

4.9.7 Verifying the Period Range

Enter the STATUS command to review the view settings and verify that a SET PERIOD is in effect.

The SET PERIOD command maintains the specified dates until you enter the SET PERIOD DEFAULT command.

4.10 Periodicities

4.10.1 About Periodicities

By default, data is displayed in a dimensional model at the periodicity specified when the variables were created in the dimensional model.

A number of default periodicity definitions are provided. In addition, you can create your own periodicity definitions, which then appear in the "User defined" section of the Calendar dialog box. If you do not create any periodicity definitions, or if the document that contains your definitions is not found, the Calendar dialog box displays only the default periodicities.

You can use the Calendar dialog box to change the periodicity at which your data is displayed. For example, if your data is stored weekly, you can select a different periodicity in the Calendar dialog box to display the data biweekly, monthly, yearly, and so on.

Note: You can select either a periodicity or time template in the Calendar dialog box, but not both together.

4.10.2 Default Periodicities

4.10.2.1 Bimonthly

Periodicity description

Displays data for each two-month period.

Example

This example shows several gross Sales values for a model with a start date of January 2, 1999:

```
Jan - Feb 99    12,875,050.00
```

Periodicities

Mar - Apr 99 11,616,017.50

May - Jun 99 11,491,437.00

4.10.2.2 Biweekly

Periodicity description

Displays data for each two-week period.

Example

This example shows several gross Sales values for a model with a start date of January 2, 1999:

01 Jan - 14 Jan 99 2,552,923.48

15 Jan - 28 Jan 99 2,736,415.00

29 Jan - 11 Feb 99 3,286,490.48

12 Feb - 25 Feb 99 3,486,554.21

4.10.2.3 Daily

Periodicity description

Displays data for each day.

Example

This example shows several gross Sales values for a model with a start date of January 2, 1999:

02 Jan 99 983,980.00

03 Jan 99 883,860.00

04 Jan 99 543,970.00

05 Jan 99 750,789.00

06 Jan 99 870,456.00

4.10.2.4 Hourly

Periodicity description

Displays data for each hour.

Example

This example shows several gross Sales values for a model with a start date of January 2, 1999:

1 am 02 Jan 980.00

2 am 02 Jan 980.00

3 am 02 Jan 980.00

4 am 02 Jan 980.00

5 am 02 Jan 980.00

6 am 02 Jan 980.00

7 am 02 Jan 980.00

8 am 02 Jan 980.00

9 am 02 Jan 980.00
10 am 02 Jan 1050.00
11 am 02 Jan 1250.00

Note: If a message is displayed, advising you that the data exceeds the maximum number of observations, request that your database administrator changes the model within Application Server.

4.10.2.5 Lunar to Date

Periodicity description

Displays data for each lunar month, up to the specified end date. Data for previous years is only displayed up to the same date as the end date. For example, if the end date is May 31, 1999, data for previous years is displayed up to May 31.

Example

This example shows several gross Sales values for a model with a start date of January 2, 1999 and an end date of May 31, 1999:

28 Jan 99	11,586,957.29
25 Feb 99	14,419,289.88
24 Mar 99	13,327,199.41
21 Apr 99	12,365,095.92
19 May 99	12,331,234.89

4.10.2.6 Monthly

Periodicity description

Displays data for each month between the specified start and end dates.

Example

This example shows gross Sales values for a model with a start date of January 2, 1999 and an end date of May 31, 1999:

Jan 99	12,828,417.00
Feb 99	15,286,283.00
Mar 99	14,490,881.00
Apr 99	12,989,945.00
May 99	13,761,147.00

4.10.2.7 Monthly to Date

Periodicity description

Displays data for each month, up to the specified end date. Data for previous years is only displayed up to the same date as the end date. For example, if the end date is May 31, 1999, data for previous years is displayed up to May 31.

Periodicities**Example**

This example shows gross Sales values for a model with a start date of January 2, 1999 and an end date of May 31, 1999:

29 Jan 99	11,586,957.00
26 Feb 99	14,532,583.19
25 Mar 99	13,267,533.98
22 Apr 99	12,330,646.31
20 May 99	12,342,144.69

4.10.2.8 Qtr Hourly

Periodicity description

Displays data for each quarter-hourly point.

Example

This example shows gross Sales values for a model with a start date of January 2, 1999:

12.00 J 2	579.60
12.15 J 2	579.60
12.30 J 2	579.60
12.45 J 2	579.60
1.00 J 2	579.60

Note: If a message is displayed, advising you that the data exceeds the maximum number of observations, request that your database administrator change the model within Application Server.

4.10.2.9 Qtrly to Date

Periodicity description

Displays data for each quarter, up to the specified end date.

Example

This example shows gross Sales values for a model with a start date of January 2, 1998 and an end date of May 31, 1999:

Jan-Mar Qtd	28,114,700.00
Apr-Jun Qtd	26,751,092.00
Jul-Sep Qtd	27,158,206.00
Oct-Dec Qtd	26,645,774.00
Jan-Mar Qtd	37,642,856.00
Apr-Jun Qtd	37,371,054.00

4.10.2.10 Quarterly

Periodicity description

Displays data for each quarter, or three month period, within the specified start and end dates.

Example

This example shows gross Sales values for a model with a start date of January 2, 1999 and an end date of May 31, 1999:

Jan-Mar 99	42,605,581.00
Apr-Jun 99	40,752,619.00

4.10.2.11 Rolling Monthly

Rolling periodicities operate independently of the nominal year, quarter, or month. You can use rolling periodicities to look at a period of time as if it were a year, quarter, or month.

Periodicity description

Displays data in four-week blocks.

Example

This example shows gross Sales values for a model with a start date of January 2, 1999:

15/01/99	834,629.33
12/02/99	1,764,098.46
11/03/99	1,914,337.46
08/04/99	1,714,422.98
06/05/99	1,626,057.88
03/06/99	1,395,092.52

4.10.2.12 Rolling Qtrly

Rolling periodicities operate independently of the nominal year, quarter, or month. You can use rolling periodicities to look at a period of time as if it were a year, quarter, or month.

Periodicity description

Displays data in three-month blocks.

Example

This example shows gross Sales values for a model with a start date of January 2, 1999:

Jan-Mar 99	5,742,911.87
Apr-Jun 99	5,056,326.75

4.10.2.13 Rolling Yearly

Rolling periodicities operate independently of the nominal year, quarter, or month. You can use rolling periodicities to look at a period of time as if it were a year, quarter, or month.

Periodicity description

Displays data in 12-month blocks.

Example

This example shows gross Sales values for a model with a start date of January 2, 1998:

1998	10,799,238.63
1999	26,361,835.88

4.10.2.14 Semi Annual

Periodicity description

Displays data for six-month periods.

Example

This example shows gross Sales values for a model with a start date of January 2, 1998 and an end date of May 31, 1999:

Jan-Jun 98	82,605,581.00
Jul-Dec 98	80,752,619.00
Jan-Jun 99	86,989,486.00

4.10.2.15 Weekly

Periodicity description

Displays data for each week.

Example

This example shows gross Sales values for January for a model with a start date of January 2, 1999:

08 Jan 99	4,833,055.00
15 Jan 99	4,083,860.00
22 Jan 99	3,543,970.00
29 Jan 99	3,780,789.00

4.10.2.16 Weekly to Date

Periodicity description

Displays data for each week, up to the specified end date.

Example

This example shows gross Sales values for January for a model with a start date of January 2, 1999:

08 Jan 99	2,058,972.77
15 Jan 99	2,058,972.77
22 Jan 99	2,058,972.77
29 Jan 99	2,058,972.77
05 Feb 99	2,474,175.35
12 Feb 99	2,640,256.38

4.10.2.17 Year to Date

Periodicity description

Displays data for each year, up to the specified end date.

Example

This example shows the gross Sales value for a model with a start date of January 2, 1999:

1999 Ytd 30,601,597.00

4.10.2.18 Yearly**Periodicity description**

Displays data for each year within the specified start and end dates.

Example

This example shows the gross Sales value for a model with a start date of January 2, 1998 and an end date of December 31, 1998:

1998 542,286,210.00

4.10.3 General Periodicities

When you create a variable, you assign a general periodicity that specifies the unit of time in which you want to store the data. For example, if you create a variable with a monthly periodicity, each data point for that variable represents data for one month.

You can display a variable in a different periodicity than its assigned periodicity. For example, if a variable has a monthly periodicity, you can display it quarterly or weekly (Application Server divides the value into 4 or 5 weeks).

Periodicity	Displays data for every:
Yearly	12 months
Semiannual	6 months
Quarterly	quarter
Bimonthly	2 months
Monthly	month
Lunar	28 days. Use this with a 13-month fiscal year only.
Weekly	week
Biweekly	2 weeks
Daily	day
Hourly	hour

4.10.4 Period-to-Date Periodicities

When you display a variable, you can specify the period-to-date at which to display data.

For example, if you set the latest date to be the end of April of the current year, and you display your data for the past three years using the YTD periodicity, you see data for those past three years only through April of each year. You can compare the current year's year-to-date data with the previous year's data.

You can display a variable in a longer periodicity than its assigned periodicity. For example, if a variable has a monthly periodicity, you can display data for quarter-to-date but you cannot display it week-to-date.

Periodicity	Displays data for the:
MTD	month-to-date
MYTD	year-to-date, by month
Note: You cannot use MYTD for quarterly or yearly variables.	

Time Templates

QTD	quarter-to-date
WTD	week-to-date
YTD	year-to-date

4.10.5 Rolling Periodicities

When you display a variable, you can use a rolling periodicity to operate independently of the nominal year, quarter, or month, and to look at some period of time as if it were a year, a quarter, or a month.

For example, if you set the latest month to be May, a Rquarterly periodicity returns data for the months of March, April, and May.

You can display a variable in a longer periodicity than its assigned periodicity. For example, if a variable has a quarterly periodicity, you can display it Rquarterly but you cannot display it Rmonthly.

Periodicity	Displays data for:
Ryearly	the most recent 12 months.
Rquarterly	the most recent 3 months.
Rmonthly	the most recent 4 weeks.

4.11 Time Templates

4.11.1 About Time Templates

You can use time templates to do the following:

Display data at more than one periodicity. For example, you can display monthly and year to date data in the view.

Compare data from different time periods. For example, you can calculate the percentage change in sales between the current month and the same month a year ago, and display the result in the view.

Change the order in which you display data.

Default time templates

Year to date templates

% Change templates

4.11.2 Defining Which Time Templates To Use

You can define which time templates to display in the Periodicity and Templates tabs of the Calendar by creating a definitions document.

When you start the Calendar, the program searches for the definitions document first in the current Application Server Use database, and then, if it is not found, in the APLIB database. If no definitions document is found in either database, or if the Calendar is launched without a connection to Application Server, the default periodicities and time templates are used that are provided with the application you are currently running.

The name of the definitions document is based on the locale setting in the format `CAL_LNG`, where `LNG` is the standard locale suffix (for example, the English version must be called `CAL_ENG`). Default versions of the `CAL_LNG` documents that define the Periodicity, Year to Date, and %

Change lists are provided in the APLIB database. To add or delete time sets from the default lists, or to create your own lists, modify the appropriate document for your locale setting.

Format of the definitions document

The definitions document must be in a specific format, as follows:

```
[
PERIOD
    [periodicity]*
]
[
CATEGORY [category]
    [shortname    longname]*
]*
```

where:

<i>periodicity</i>	specifies the internal Application Server periodicities
<i>category</i>	specifies the category name
<i>shortname</i>	specifies the time set short name as it exists in the database
<i>longname</i>	specifies the descriptive name to display for the template in the Calendar
[]*	indicates a syntax that may be repeated

The PERIOD section of the definitions document is used to customize the list of built-in Application Server periodicities that are displayed in the Periodicity list on the Periodicity tab of the Calendar.

Within the document, you can specify up to 20 CATEGORY sections, each containing a separate list of templates that can be selected from the Templates list on the Template tab of the Calendar. You can also include time templates that reside in other attached databases (see Example 3).

The category name is separated from the CATEGORY keyword by one or more spaces. You must specify the time template short name and long name for each entry, separated by a TAB character.

In addition to templates defined in the document, user-defined time templates that reside in your Work and Use databases appear in the Calendar Templates list as "Work database" and "Use database".

4.11.3 User-Defined Time Templates

You now have more flexibility using time templates in the Calendar. You can define the behavior of the Calendar from each time template.

By using the following keyword as the first line of a user-defined time template in your Application Server Use or Work database, you can control several aspects of how the Calendar functions. You precede this line with ellipses (...):

...TemplateDates={Latest|Both}

The following Calendar options and Application Server commands issued by the Calendar, are controlled by the content of TemplateDates in all time templates used by the Calendar:

Start Date field in the Calendar (active or inactive)

SET PERIOD command sent to Application Server

SET EARLIEST command sent to Application Server

SET LATEST command sent to Application Server

Time Templates

The following table describes the actions and options performed by the Calendar depending on the value of TemplateDates:

TemplateDates value	Start Date field	Set Period value	Set Earliest value	Set Latest value
Latest	inactive	Default	start date	end date
Both	active	start-end	start date	end date

Defining an alternative long description for a template

You can now use the "...Description=*description*" specification to define an alternative long description for a user-defined template found in the Work or Use databases. For example:

...TemplateDates=Latest

...Description=Favorite view

4.11.4 Default Time Templates

4.11.4.1 Year to Date Templates

4.11.4.1.1 Year to Date Template Short Names

This table lists the default year to date time templates and their short names. For more information about default time templates, click an underlined topic.

Template long name	Template short name
Last 12 Months Ytd	CMTYD
Latest Day v Ytd	DAYYTD
Latest Mtd v Prior	MTDLASTMTD
Latest Mtd v Yr Ago	MTDLASTYEARMTD
Latest Qtd v Prior	QTDLASTQTD
Latest Qtd v Yr Ago	QTDLASTYEARQTD
Latest Wtd v Prior	WTDLASTWTD
Latest Wtd v Yr Ago	WTDLASTYEARWTD
Latest Ytd v Prior	YTDLASTYTD
Mon/Prior, Yr/Prior	MONPMONYRPPYR
Mon/Yr v Prior	MONYRMONPYR
Month/Ytd	MONTHYTD
Q1-Q4/Ytd	QQQQTYD
Quarter/Ytd	QTRYTD
Week/Ytd	WEEKYTD
Ytd	ONLYYTD

Note: The default time templates are contained in the APLIB database.

4.11.4.1.2 Last 12 Months Ytd

Template description

Displays up to 12 columns of monthly data on a year to date (YTD) basis, ending with the most current month.

Template short name

CMYTD

Example

This example shows gross Sales for a model with a latest date of May 28, 1999:

Jan 99 Ytd	68,339,832.00
Feb 99 Ytd	139,596,952.00
Mar 99 Ytd	215,210,480.00
Apr 99 Ytd	290,921,808.00
May 99 Ytd	371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.3 Latest Day v Ytd

Template description

Displays the latest daily data point and the year to date (YTD) value through that point.

Template short name

DAYYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

5/28	2,883,860.00
Ytd	371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.4 Latest Mtd v Prior

Template description

Displays the latest month period on a month to date (MTD) basis compared with the preceding month. Do not use this template unless your data is stored in periods smaller than monthly.

Template short name

MTDLASTMTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 99 Mtd	80,748,080.00
Apr 99 Mtd	75,711,328.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.5 Latest Mtd v Yr Ago

Template description

Displays the latest month period on a month to date (MTD) basis compared with the same period one year ago. Do not use this template unless your data is stored in periods smaller than monthly.

Template short name

MTDLASTYEARMTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 99 Mtd 80,748,080.00

May 98 Mtd 45,888,920.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.6 Latest Qtd v Prior

Template description

Displays the current quarter to date (QTD) compared with the preceding quarter to date.

Template short name

QTDLASTQTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Apr-Jun Qtd 156,459,408.00

Jan-Mar Qtd 139,596,952.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.7 Latest Qtd v Yr Ago

Template description

Displays the current quarter to date (QTD) compared with the same period one year ago.

Template short name

QTDLASTYEARQTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Apr-Jun Qtd 156,459,408.00

Apr-Jun Qtd 91,526,720.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.8 Latest Wtd v Prior

Template description

Displays the latest week to date (WTD) compared with the preceding period.

Template short name

WTDLASTWTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99	20,187,020.00
21 May 99	20,187,020.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.9 Latest Wtd v Yr Ago

Template description

Displays the latest week to date (WTD) compared with the same period one year ago.

Template short name

WTDLASTYEARWTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99	20,187,020.00
29 May 98	11,472,230.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.10 Latest Ytd v Prior

Template description

Displays the latest year to date (YTD) data compared with the data from one year ago.

Template short name

YTDLASTYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

1999 Ytd	371,669,888.00
1998 Ytd	217,624,872.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.11 Mon/Prior, Yr/Prior

Template description

Displays values for the latest month and the same month in the previous year, and year to date values for the current and previous years.

Template short name

MONPMONYRPPYR

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 1999	80,748,080.00
May 1998	45,888,920.00
1999 Ytd	371,669,888.00

Time Templates

1998 Ytd	217,624,872.00
----------	----------------

Note: The default time templates are contained in the APLIB database.

4.11.4.1.12 Mon/Yr v Prior

Template description

Displays the latest month and year to date values for the current year and the previous year.

Template short name

MONYRPMONPYR

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 1999	80,748,080.00
1999 Ytd	371,669,888.00
May 1998	45,888,920.00
1998 Ytd	217,624,872.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.13 Month/Ytd

Template description

Displays the latest month and year to date values for the current year.

Template short name

MONTHYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 1999	80,748,080.00
1999 Ytd	371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.14 Q1-Q4/Ytd

Template description

Displays quarterly and year to date values for the previous year.

Template short name

QQQQYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Jan-Mar 98	126,098,152.00
Apr-Jun 98	139,149,292.00
Jul-Sep 98	158,062,580.00

Oct-Dec 98 178,320,816.00

1998 601,630,840.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.15 Quarter/Ytd

Template description

Displays the latest quarter and year to date values for the current year.

Template short name

QTRYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Apr-Jun 1999 238,981,616.00

1999 Ytd 371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.16 Week/Ytd

Template description

Displays the latest week and year to date values for the current year.

Template short name

WEEKYTD

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99 20,187,020.00

1999 Ytd 371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.1.17 Ytd

Template description

Displays the latest year to date value for the current year.

Template short name

ONLYYTD

Example

This example shows the gross Sales value for a model with a latest date of May 28, 1999:

1999 Ytd 371,669,888.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2 % Change Templates

4.11.4.2.1 % Change Template Short Names

This table lists the default % change time templates and their short names. For more information about default time templates, click an underlined topic.

Template long name	Template short name
Latest 4 Quarters	QRTRLY
Latest 12 Month	MNTHLY
Latest 13 Weeks	WKLY1
Latest 14 Days	DLY
Latest Day v Yr Ago	DAYLY
Latest Month v Prior	MONLASTMON
Latest Month v Yr Ago	MONLASTYEARMON
Latest Qtr v Prior	QUALASTQUA
Latest Qtr v Yr Ago	QUALASTYEARQUA
Latest Semi v Prior	SEMILY
Latest Week v Prior	WEELASTWEE
Latest Week v Yr Ago	WEELASTYEARWEE
Latest Year v Prior	YEALASTYEA
Lunar Latest v Prior	LUNLASTLUN
Lunar Latest v Yr Ago	LUNLASTYEARLUN
Month v Prior/% Chg	MTHLY
Prior Mth v Var	VLV
Qtr v Prior/% Chg	QTRLY
Week v Prior/% Chg	WKLY
Year v Prior/% Chg	YRLY
Ytd v Prior/% Chg	YTDLY

Note: The default time templates are contained in the APLIB database.

4.11.4.2.2 Latest 4 Quarters

Template description

Displays the last four quarters, though the current period.

Template short name

QRTRLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Jul-Sep 98	158,062,580.00
Oct-Dec 98	178,320,816.00
Jan-Mar 99	215,210,480.00
Apr-Jun 99	238,981,616.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.3 Latest 12 Month

Template description

Displays the last 12 months of data, through the current month.

Template short name

MNTHLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Jun 98 47,622,572.00

Jul 98 49,327,344.00

Aug 98 51,919,672.00

Sep 98 56,815,564.00

Oct 98 57,681,632.00

Nov 98 59,990,968.00

Dec 98 60,648,216.00

Jan 99 68,339,832.00

Feb 99 71,257,120.00

Mar 99 75,613,528.00

Apr 99 75,711,328.00

May 99 80,748,080.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.4 Latest 13 Weeks

Template description

Displays the last 13 weeks of weekly data.

Template short name

WKLY1

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

05 Mar 99 15,122,705.60

12 Mar 99 15,122,705.60

19 Mar 99 15,122,705.60

26 Mar 99 15,122,705.60

02 Apr 99 15,122,705.60

09 Apr 99 18,927,832.00

16 Apr 99 18,927,832.00

23 Apr 99 18,927,832.00

Time Templates

30 Apr 99	18,927,832.00
07 May 99	20,187,020.00
14 May 99	20,187,020.00
21 May 99	20,187,020.00
28 May 99	20,187,020.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.5 Latest 14 Days

Template description

Displays the last 14 days of daily data.

Template short name

DLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

15 May 99	2,883,860.00
16 May 99	2,883,860.00
17 May 99	2,883,860.00
18 May 99	2,883,860.00
19 May 99	2,883,860.00
20 May 99	2,883,860.00
21 May 99	2,883,860.00
22 May 99	2,883,860.00
23 May 99	2,883,860.00
24 May 99	2,883,860.00
25 May 99	2,883,860.00
26 May 99	2,883,860.00
27 May 99	2,883,860.00
28 May 99	2,883,860.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.6 Latest Day v Yr Ago

Template description

Displays the latest daily data point compared with the same point one year ago.

Template short name

DAYLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

5/28/99 2,883,860.00

5/28/98 1,638,890.00

% Change 75.96

Note: The default time templates are contained in the APLIB database.

4.11.4.2.7 Latest Month v Prior

Template description

Displays the latest month compared with the preceding month.

Template short name

MONLASTMON

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 99 80,748,080.00

Apr 99 75,711,328.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.8 Latest Month v Yr Ago

Template description

Displays the latest month compared with the same month last year.

Template short name

MONLASTYEARMON

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 99 80,748,080.00

May 98 45,888,920.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.9 Latest Qtr v Prior

Template description

Displays the latest quarter compared with the preceding quarter.

Template short name

QUALASTQUA

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Apr-Jun 99 238,981,616.00

Jan-Mar 99 215,210,480.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.10 Latest Qtr v Yr Ago

Template description

Displays the latest quarter compared with the same period one year ago.

Template short name

QUALASTYEARQUA

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Apr-Jun 99	238,981,616.00
------------	----------------

Apr-Jun 98	139,149,292.00
------------	----------------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.11 Latest Semi v Prior

Template description

Displays the latest half year period compared with the preceding period, and shows the percentage change between them.

Template short name

SEMILY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Half1 99	454,192,096.00
----------	----------------

Half1 98	265,247,444.00
----------	----------------

% Change	71.23
----------	-------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.12 Latest Week v Prior

Template description

Displays the latest weekly period compared with the preceding period.

Template short name

WEELASTWEE

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99	20,187,020.00
-----------	---------------

21 May 99	20,187,020.00
-----------	---------------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.13 Latest Week v Yr Ago

Template description

Displays the latest weekly period compared with the same period one year ago.

Template short name

WEELASTYEARWEE

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99	20,187,020.00
-----------	---------------

29 May 98	11,472,230.00
-----------	---------------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.14 Latest Year v Prior

Template description

Displays the data for the latest year compared with the data from one year ago.

Template short name

YEALASTYEA

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

1999	985,564,392.00
------	----------------

1998	601,630,840.00
------	----------------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.15 Lunar Latest v Prior

Template description

Displays the current Lunar to Date period compared with the preceding lunar (28 day) period.

Template short name

LUNLASTLUN

Example

This example shows Sales values for a model with a latest date of December 3, 1998:

03 Dec 98	537,975.00
-----------	------------

05 Nov 98	589,069.00
-----------	------------

Note: The default time templates are contained in the APLIB database.

4.11.4.2.16 Lunar Latest v Yr Ago

Template description

Displays the current lunar period compared with the same period one year ago.

Template short name

LUNLASTYEARLUN

Example

This example shows Sales values for a model with a latest date of December 3, 1998:

03 Dec 98 537,975.00

03 Dec 97 518,654.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.17 Month v Prior/% Chg

Template description

Displays values for the latest month, the same month in the previous year, and the percentage change between the values.

Template short name

MTHLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 99 80,748,080.00

May 98 45,888,920.00

% Change 76.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.18 Prior Mth v Var

Template description

Compares the latest period to the same period one year ago, but displays Prior Yr followed by the variance of this year to last year.

Template short name

VLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

May 97 45,888,920.00

Variance 34,859,160.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.19 Qtr v Prior/% Chg

Template description

Displays values for the latest quarter, the same quarter in the previous year, and the percentage change between the values.

Template short name

QTRLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

Qtr 2 99	238,981,616.00
Qtr 2 98	139,149,292.00
% Change	72.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.20 Week v Prior/% Chg

Template description

Displays values for the latest week, the same week in the previous year, and the percentage change between the values.

Template short name

WKLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

28 May 99	20,187,020.00
29 May 98	11,472,230.00
% Change	76.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.21 Year v Prior/% Chg

Template description

Displays values for the current and previous years, and the percentage change between the values.

Template short name

YRLY

Example

This example shows gross Sales values for a model with a latest date of December 28 1998:

1998	985,564,392.00
1997	601,630,840.00
% Change	64.00

Note: The default time templates are contained in the APLIB database.

4.11.4.2.22 Ytd v Prior/% Chg

Template description

Displays year to date (Ytd) values for the current and previous years, and the percentage change between the values.

Template short name

YTDLY

Example

This example shows gross Sales values for a model with a latest date of May 28, 1999:

1999 Ytd	371,669,888.00
1998 Ytd	217,624,872.00
% Change	71.00

Note: The default time templates are contained in the APLIB database.

4.11.5 Fiscal Patterns

All fiscal patterns, such as 445 or 454, always begin in the month of January, even if you specify a month other than January to begin the fiscal year.

Patterns	454	445	544
January	4	4	5
February	5	4	4
March	4	5	4
April	4	4	5
May	5	4	4
June	4	5	4
July	4	4	5
August	5	4	4
September	4	5	4
October	4	4	5
November	5	4	4
December	4	5	4

To begin a fiscal year in August and use a 544 pattern in which August has five weeks, September has four weeks, and October has four weeks, you would use a command similar to this:

SET FISCAL User 454 First August

See also

SET FISCAL

4.12 User-Defined Hierarchies

4.12.1 About User-Defined Hierarchies

A User-Defined Hierarchy is a selection of a dimension's members that you identify by a name. Once you define a User-Defined Hierarchy, you can use it to select and display the User-Defined Hierarchy without having to specify each member.

You can define multiple User-Defined Hierarchies for a dimension, and you can include User-Defined Hierarchies in a User-Defined Hierarchy.

Note: When switching databases, all selections related to that database, such as User-Defined Hierarchy creations, are discarded.

Overview of working with User-Defined Hierarchies

1. Enable a new dimension or an existing dimension to have User-Defined Hierarchies.
2. Create User-Defined Hierarchies for the dimension.
3. Once you have created a User-Defined Hierarchy for a dimension, you can:
 - Select the User-Defined Hierarchy rather than select each member of the User-Defined Hierarchy. Application Server bases the Across and Down view on the User-Defined Hierarchy you select. Application Server automatically aggregates the members.
 - Show details of the User-Defined Hierarchy.
 - List the User-Defined Hierarchies or the members of the User-Defined Hierarchy.
 - Clear all or individual User-Defined Hierarchies from a dimension.
 - Use the User-Defined Hierarchy in this session or save it to use in future sessions.

Note: For User-Defined Hierarchies to work, the variables involved must have rollups defined.

4.12.2 Creating User-Defined Hierarchies

4.12.2.1 Enabling a New Dimension To Have User-Defined Hierarchies

Procedure

1. Create a procedure by entering:

PROCEDURE <setname>

where <setname> is the name of the procedure.

2. In the Procedure editor, specify the appropriate ACCESS subsystem, the source file, and descriptions for the dimension:

ACCESS External

ACCESS Link

USE <filename>

CONNECT name

BEGIN

SELECT <field>, <field>, ... **FROM**

DESCRIPTION

<filename>

statements ...

END

3. Construct the dimension using the CUSTOM statement:

CONSTRUCT <dimension> **LEVEL** <field>, ..., <field> **PREFACE** "CUSTOM <number>"

where <dimension> is the name of the dimension to construct.

where <fields> are the source field names that you are constructing as dimension levels. You can specify multiple fields, separated by commas, to construct multiple levels for the dimension.

where <number> is the maximum number of User-Defined Hierarchies that can be defined for this dimension.

4. Add the following lines to exit from the Access subsystem and compile the dimension:

END
COMPILE DIMENSION <dimension>

where <dimension> is the name of the dimension you are constructing.

Note: When you re-compile a dimension, the User-Defined Hierarchy information that Application Server inserts into a copy of the compiled dimension in the work database is cleared.

5. Save the changes to the procedure and exit from the Procedure editor by choosing Exit from the File menu.
6. Run the procedure by entering:

JOB <setname>

Application Server constructs the dimension with the ability to have User-Defined Hierarchies. CUSTOM GROUPS *number* appears as the first line in the dimension set.

Note: For User-Defined Hierarchies to work, the variables involved must have rollups defined.

Note: When you compile a dimension (either standard dimension or one used for a User-Defined Hierarchy), the system automatically searches for multiple counting issues that arise if a dimension's members have multiple parents. During aggregation, the system corrects multiple counting issues by creating adjustments to eliminate the multiple counts. The adjustments are used during a CONSOLIDATE. No data values are changed until a CONSOLIDATE command is issued to reconsolidate data.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC1 Selected

The dimension compiler detected multiple counting issues in Dimension PRODUCT.

Adjustments have been made to internal aggregation rules to correct these issues.

Automatic multiple counting corrections can only be performed on measures where a smart fast consolidate can be used (i.e., all of the dimensions have either simple additive or subtractive consolidations) – i.e. the measure can be used with the ROLLUP editor.

Automatic multiple counting corrections will not be performed on a measure if:

- Any of its dimensions have non additive calculations
- The measure is an INTEGRAL measure
- The user uses the NOCORRECTIONS keyword on COMPILE DIMENSION. This turns off the automatic multiple counting corrections for that dimension.
- All aggregations of a member that cause multiple counting must be of the same "sign" i.e. all additive or all subtractive.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC3 Selected

Dimension PRODUCT contains both positive and negative sums.

Compiler could not check for multiple counting

4.12.2.2 Enabling an Existing Dimension To Have User-Defined Hierarchies

Procedure

1. Start the dimension editor by entering:

DIMENSION <dimension>

where <dimension> is the name of the dimension for which you are defining User-Defined Hierarchies.

2. Add the CUSTOM statement to the first line:

CUSTOM <number>

where <number> is the number of User-Defined Hierarchies that can be defined for the dimension.

3. Save the changes and exit the dimension editor.

Application Server automatically compiles the dimension set with the changes when you exit.

Note: For User-Defined Hierarchies to work, the variables involved must have rollups defined.

Note: When you compile a dimension (either standard dimension or one used for a User-Defined Hierarchy), the system automatically searches for multiple counting issues that arise if a dimension's members have multiple parents. During aggregation, the system corrects multiple counting issues by creating adjustments to eliminate the multiple counts. The adjustments are used during a CONSOLIDATE. No data values are changed until a CONSOLIDATE command is issued to reconsolidate data.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC1 Selected

The dimension compiler detected multiple counting issues in Dimension PRODUCT.

Adjustments have been made to internal aggregation rules to correct these issues.

Automatic multiple counting corrections can only be performed on measures where a smart fast consolidate can be used (i.e., all of the dimensions have either simple additive or subtractive consolidations) – i.e. the measure can be used with the ROLLUP editor.

Automatic multiple counting corrections will not be performed on a measure if:

- Any of its dimensions have non additive calculations
- The measure is an INTEGRAL measure
- The user uses the NOCORRECTIONS keyword on COMPILE DIMENSION. This turns off the automatic multiple counting corrections for that dimension.
- All aggregations of a member that cause multiple counting must be of the same "sign" i.e. all additive or all subtractive.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC3 Selected

Dimension PRODUCT contains both positive and negative sums.

Compiler could not check for multiple counting

4.12.2.3 Creating User-Defined Hierarchies for a Dimension

Once you enable a dimension to have User-Defined Hierarchies, the CUSTOM GROUPS statement appears as the first line in the dimension set. You can create User-Defined Hierarchies for that dimension, up to the number specified in the CUSTOM GROUPS statement.

To create a User-Defined Hierarchy for a dimension, enter any combination of dimension members (excluding result members, level names, or class names) or previously defined User-Defined Hierarchies, or use the current selections:

```
CREATE <dimension> <user_defined_hierarchy> [ "<label>" ] = { SELECTED }
    { <member> | <user_defined_hierarchy> } [+|- <member> | <user_defined_hierarchy> ... ]
```

or

```
CREATE <dimension> <user_defined_hierarchy> [ "<label>" ] = <member> | <user_defined_hierarchy>
    + | - <member> | <user_defined_hierarchy> ...
```

where <dimension> is the name of the dimension for which you are creating User-Defined Hierarchies.

where <user_defined_hierarchy> is the name of the user_defined_hierarchy you want to create.

where <member> is the member of the dimension to be included in the User-Defined Hierarchy. Use the plus (+) and minus (-) signs to add or subtract members from a User-Defined Hierarchy.

where <label> is the label for the User-Defined Hierarchy member. The labels must be in double (") quotation marks. Use up to 250 characters.

where <user_defined_hierarchy> is the User-Defined Hierarchy to be included in the User-Defined Hierarchy. Use the plus (+) and minus (-) signs to add or subtract User-Defined Hierarchies.

SELECTED includes all selected members of the dimension in the User-Defined Hierarchy.

Notes:

For User-Defined Hierarchies to work, the variables involved must have rollups defined.

A User-Defined Hierarchy is created in the level above its highest members. For example, if all members of a User-Defined Hierarchy are input members, the User-Defined Hierarchy is created at the first output level.

4.12.3 Using User-Defined Hierarchies

4.12.3.1 Selecting a User-Defined Hierarchy

When you select a User-Defined Hierarchy, Application Server aggregates all the members of the User-Defined Hierarchy according to the rules in the User-Defined Hierarchy's definition. That is because a User-Defined Hierarchy behaves like any other output member. Therefore, what you are selecting is the result of that aggregation.

Procedure to select a User-Defined Hierarchy

Enter:

```
SELECT <dimension> <user_defined_hierarchy>
```

Procedure to select only the members of the User-Defined Hierarchy

Enter:

```
SELECT <dimension> ONLY JUST BELOW <user_defined_hierarchy>
```


Procedure to select the members of the User-Defined Hierarchy as well as the User-Defined Hierarchy output

Enter:

```
SELECT <dimension> JUST BELOW <user_defined_hierarchy>
```

Procedure to select all the User-Defined Hierarchies for a dimension

Enter:

```
SELECT <dimension> CUSTOMGROUPS.
```

Procedure to select a dimension's levels and exclude User-Defined Hierarchies from the selection

Enter:

```
SELECT <dimension> LEVEL <level> MINUS CUSTOMGROUPS
```

4.12.3.2 Listing a Dimension's User-Defined Hierarchies

Enter:

```
EXHIBIT CUSTOM <dimension>
```

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to list.

Procedure to exhibit a dimension's selected User-Defined Hierarchies

Enter:

```
EXHIBIT CUSTOM <dimension> SELECTED
```

where <dimension> is the name of a dimension whose User-Defined Hierarchies you want to view.

4.12.3.3 Saving a Dimension's User-Defined Hierarchies

You can save a dimension's User-Defined Hierarchies to use in the next Application Server session. If you do not save the User-Defined Hierarchies, Application Server clears them when you enter the EXIT CLEAR command.

Procedure to save a dimension's User-Defined Hierarchies that can be quickly restored in the next session

Enter:

```
SAVE CUSTOM <dimension> <owner>.<setname> [PUBLIC | PRIVATE]
```

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to save.

where <owner> is the Application Server user who created the User-Defined Hierarchy. If you omit the owner name, it is saved to the current user name.

where <setname> is the name of the procedure in which to save the User-Defined Hierarchy information.

where PUBLIC or PRIVATE specifies who will gain access to this User-Defined Hierarchy. You can make it available to all users, or to only the user who created the User-Defined Hierarchy.

Note: To restore the User-Defined Hierarchies in the next session, you would have to issue the RESTORE command. The User-Defined Hierarchies will be restored from the compiled dimension

sets saved in the CGLIB database. This method allows you to quickly restore and retrieve User-Defined Hierarchies.

Procedure to save a dimension's User-Defined Hierarchies that can be completely recreated in the next session

Enter:

SAVE CUSTOM <dimension> <setname>

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to save.

where <setname> is the name of the procedure in which to save the User-Defined Hierarchy information.

Note: To restore the User-Defined Hierarchies in the next session, you would have to execute the procedure set. The User-Defined Hierarchies will be recreated from the definitions in the procedure set.

4.12.3.4 Restoring User-Defined Hierarchies for the Next Session

Procedure

1. (Optional) Use the SHOW CUSTOM command to list the set names of the User-Defined Hierarchies you want to restore. You can use wild cards in this syntax:

SHOW CUSTOM TABS <database>.<owner>.<dimension>.<setname>

where TABS formats the output as tab-separated.

where <database> is the name of the database containing the dimension. If you omit the database, the current USE database is used.

where <owner> is the Application Server user who is responsible for the User-Defined Hierarchy. This owner was defined during the SAVE CUSTOM command. If you omit this name, the current Application Server user is used.

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to show.

where <setname> is the User-Defined Hierarchy procedure to show. This name was specified during the SAVE CUSTOM command.

You can use wildcards in any part of the <database>.<owner>.<dimension>.<setname> syntax.

2. Use the RESTORE CUSTOM command to restore the User-Defined Hierarchies for the session. Enter:

RESTORE CUSTOM <dimension> <owner>.<setname>

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to restore in the current session.

where <owner> is the Application Server user who created the User-Defined Hierarchy. If you omit the owner name, the current user name is used.

where <setname> is the name of the procedure that contains the User-Defined Hierarchy information you want to restore.

Note: Use this procedure when the User-Defined Hierarchy was originally saved using the SAVE CUSTOM <dimension> <owner>.<setname> PUBLIC | PRIVATE command. This method restores the User-Defined Hierarchy based on the saved compiled dimension set in CGLIB. This method is

quicker than recreating User-Defined Hierarchies and allows you to retrieve User-Defined Hierarchy information quickly too.

4.12.3.5 Re-creating User-Defined Hierarchies for the Next Session

Execute the procedure set for the User-Defined Hierarchy. For example:

JOB <setname>

Note: Use this procedure when the User-Defined Hierarchy was originally saved using the SAVE CUSTOM <dimension> <setname> command. This procedure completely recreates the User-Defined Hierarchy.

4.12.4 Maintaining User-Defined Hierarchies

4.12.4.1 Showing User-Defined Hierarchy Details

Enter:

SHOW CUSTOM TABS <database>.<owner>.<dimension>.<setname>

where TABS formats the output as tab-separated.

where <database> is the name of the database containing the dimension. If you omit the database, the current USE database is used.

where <owner> is the Application Server user who is responsible for the User-Defined Hierarchy. This owner was defined during the SAVE CUSTOM command. If you omit this name, the current Application Server user is used.

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to show.

where <setname> is the User-Defined Hierarchy procedure to show. This name was specified during the SAVE CUSTOM command.

You can use wildcards in any part of the <database>.<owner>.<dimension>.<setname> syntax.

4.12.4.2 Updating a User-Defined Hierarchy

Procedure

1. Create the User-Defined Hierarchy using the CREATE <dimension> <user_defined_hierarchy> command.
2. Save the User-Defined Hierarchy using the SAVE CUSTOM <dimension> <setname> [PRIVATE | PUBLIC] CHANGE command.
3. Update the User-Defined Hierarchy using the CREATE <dimension> REPLACE <user_defined_hierarchy> command.
4. Save the updates using THE SAVE CUSTOM CHANGE command.

4.12.4.3 Clearing a Dimension's User-Defined Hierarchies

Procedure to clear all User-Defined Hierarchies from a dimension

Enter:

CLEAR <dimension>

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to clear.

Procedure to clear a dimension's User-Defined Hierarchies

Enter:

CLEAR CUSTOM <dimension> [, <dimension>...]

where <dimension> is the name of dimension whose User-Defined Hierarchies you want to clear. You can specify more than one dimension, where each one is separated by a comma.

Procedure to clear an individual User-Defined Hierarchy from a dimension

Enter:

CLEAR <dimension> <user_defined_hierarchy>

where <dimension> is the name of dimension whose User-Defined Hierarchies you want to clear.

where <user_defined_hierarchy> is the name of the User-Defined Hierarchy you want to delete. You can specify more than one User-Defined Hierarchy, where each one is separated by a comma.

Procedure to clear a dimension of its User-Defined Hierarchies that has been saved to a public or private set in the CGLIB database

Enter:

CLEAR <dimension> USING <owner>.<setname>

where <dimension> is the name of dimension whose User-Defined Hierarchies you want to clear.

where <owner> is the name of the Application Server who is responsible for the User-Defined Hierarchy. If you omit the owner, then the current Application Server user is used.

where <setname> is the name of the set created during a SAVE CUSTOM command with the PRIVATE or PUBLIC keyword.

Notes:

When you clear a dimension's User-Defined Hierarchies, Application Server selects all the dimension's members.

If the User-Defined Hierarchy has been saved using the SAVE CUSTOM command that does not include the PUBLIC or PRIVATE keywords, it can be recreated during another session by executing the procedure set.

If this User-Defined Hierarchy has been saved using the SAVE CUSTOM command that includes the PUBLIC or PRIVATE keywords, then the compiled dimension set for this User-Defined Hierarchy remains saved in the CGLIB database and will be restored again during the next RESTORE CUSTOM command.

If you want to completely remove the User-Defined Hierarchy from your database, including the compiled dimension set, then use the REMOVE CUSTOM command instead.

4.12.4.4 Removing User-Defined Hierarchies from a Dimension

Enter:

REMOVE CUSTOM <database>.<owner>.<dimension>.<setname>

where <database> is the name of the database containing the dimension. If you omit the database, the current USE database is used.

where <owner> is the Application Server user who is responsible for the User-Defined Hierarchy. This owner was defined during the SAVE CUSTOM command. If you omit this name, the current Application Server user is used.

where <dimension> is the name of the dimension whose User-Defined Hierarchies you want to delete.

where <setname> is the User-Defined Hierarchy procedure to delete. This name was specified during the SAVE CUSTOM command.

Notes:

You can remove more than one User-Defined Hierarchy at a time by separating each database.owner.dimension.setname with a comma.

At a minimum, you must specify at least the dimension name and setname.

4.13 Security and Access

4.13.1 About Security and Access

Application Server implements security using access keys:

Read access key — You can add a read access key to a database. To enable a user to read from the database, a supervisor must add the read access key to the user's record.

Update access key — You can add an update access key to a database. To enable a user to update the database, a supervisor must add the update access key to the user's record.

Dimension access — You can specify which dimension members each user or user group can access.

In addition, you can specify the maximum security level at which any user can access a database and specify a protection key to prevent unauthorized users from copying or moving the database.

Once a database is opened for a particular level of access, all users must access the database at the same level. The following database access modes are available:

Read — Multiple users can read from the database, but cannot update it.

Shared — Multiple users can read from the database, and they can update report sets, time sets, documents, and procedures. However, users cannot update data or change the structure of the database by editing dimension sets.

Exclusive — One user can read from database, and update sets and data within it.

4.13.2 Database Level

4.13.2.1 Adding a Read or Updated Access Key to a Database

Procedure to add a read access key to a database

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR CHANGE DATABASE *database* READ *read*

where:

- *database* is the name of the database to which you want to add the access key
- *read* is the access key you want to use

To enable a user to read from the database, you must add the *read* key to the user's record in MASTERDB.

Procedure to add an update access key to a database

3. 1. Access Application Server as a user with Supervisor privileges.

4. 2. Enter the following command:

SUPERVISOR CHANGE DATABASE *database* UPDATE *update*

where:

- *database* is the name of the database to which you want to add the access key
- *update* is the access key you want to use

To enable a user to update the database, you must add the *update* key to the user's record in MASTERDB.

4.13.2.2 Removing a Read or Update Access Key from a Database

Procedure

1. Access Application Server as a user with Supervisor privileges.

2. Enter the following command:

SUPERVISOR REMOVE ACCESS *access* DATABASE *database*

where:

- *access* is the access key you want to remove
- *database* is the name of the database from which you want to remove the access key

Note: Before removing an access key, you must ensure that all users are detached from the database.

4.13.2.3 Adding a Protection Key to a Database

Procedure

1. Access Application Server as a user with Supervisor privileges.

2. Enter the following command:

SUPERVISOR CHANGE DATABASE *database* PROTECT *key*

where:

- *database* is the name of the database to which you want to add the protection key
- *key* is the protection key you want to add

Notes:

You can also set a protection key on a database record when you use the CREATE DATABASE command.

Use protection keys with caution — you need to know the key name in order to remove it later (there is no way to remove an unknown protection key).

4.13.2.4 Removing a Protection Key from a Database

Procedure

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR REMOVE PROTECTION DATABASE *database key*

where:

- *database* is the name of the database from which you want to remove the protection key
- *key* is the protection key you want to remove

Notes:

Before removing a protection key, you must ensure that all users are detached from the database.

You need to know the key name in order to remove it — there is no way to remove an unknown protection key.

4.13.2.5 Specifying the Level of Access for a Database

Procedure to specify the default level of access for a database

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR CHANGE DATABASE *database* USAGE *usage*

where:

- *database* is the name of the database you want to change
- *usage* is the default access mode in which the user is attached to the database. You can specify READ for read access, SHARED for shared access, or EXCLUSIVE for exclusive access.

Note: The default access mode is used when a user enters a USE or ATTACH command without specifying a *usage* parameter.

Procedure to specify the maximum level of access for a database

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR CHANGE DATABASE *database* MAXACCESS *maxaccess*

where:

- *database* is the name of the database you want to change
- *maxaccess* is the maximum level at which a user can access the database. You can specify READ for read access or UPDATE for update access.

Note: If you specify READ for the *maxaccess* parameter, all users are denied shared and exclusive access to the database.

4.13.2.6 Specifying the Level of Access for a User

Procedure to specify the default level of access for a user

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR CHANGE USER *user* USAGE *usage*

where:

- *user* is the name of the user whose access level you want to change
- *usage* is the level of access the user gets when logging in. You can specify READ for read access, SHARED for shared access, or EXCLUSIVE for exclusive access.

Note: A user's default level of access is only used when logging in, and therefore applies to the user's default database only.

Procedure to enable a user to access a database with an access key

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR CHANGE USER *user* ACCESS *access*

where:

- *user* is the name of the user whose access rights you want to change
- *access* is the read or update access key required by the database

4.13.2.7 Revoking a User's Access to a Database

Procedure

1. Access Application Server as a user with Supervisor privileges.
2. Enter the following command:

SUPERVISOR REMOVE ACCESS *access* USER *user*

where:

- *access* is the read or update access key required by the database you want to prevent a user from accessing
- *user* is the name of the user whose access rights you want to change

4.13.3 Dimension Level

4.13.3.1 Specifying the Dimension Members Available to Each User and User Group

Procedure

1. Access Application Server as a user with the same name as the database you want to modify.
2. Enter the following command to create a procedure named Security:

PROCEDURE Security

3. Use the INDEX USER command to specify the dimension members each user and user group can access.

4. From the Procedure menu, choose Save.
5. From the Procedure menu, choose Exit.
6. Enter the following command to activate the dimension restrictions:

JOB Security

Note: The dimension restrictions do not take effect until a user detaches from and reattaches to the database. Logging out by typing Exit or choosing Exit from the File menu is not sufficient to detach the user; the user must enter an EXIT CLEAR or DETACH &USEDATABASE command.

4.14 Procedure editor

4.14.1 Editing or Creating a Procedure

Enter the PROCEDURE command following by the name of the procedure you want to edit or create. For example:

PROCEDURE SALESPROC

If a procedure with this name already exists, it is copied into the editor. If it does not exist, the procedure editor window appears. You can create the procedure.

Note: Set names can be 24 characters long and must begin with an alphabetic character. A set name cannot contain spaces unless it is enclosed in single quotation marks (' ').

The editor has four menus:

Use the Procedure menu to save your changes, quit the editor without saving changes, and so on.

Use the Edit menu to cut, copy, and paste changes.

Use the Search menu to find and replace text.

Use the Fonts menu to select fixed or proportional fonts.

Note: If you choose File Exit to quit the editor, the editor attempts to compile the set. However, certain procedures, such as those that contain control variables, or enter an Application Server subsystem, cannot be compiled. If you attempt to compile such procedures, or choose File Exit, you will receive an error; choose File Save and then File Abandon Changes instead.

4.14.2 Running a Procedure

Use one of the following three ways to run a procedure set:

JOB <set name>

<set name>

EXECUTE <set name>

Notes:

Use JOB <set name> when you are in test mode (debugging). JOB <set name> displays commands and subsequent reassurance or error messages, or both generated as each command is executed. If the procedure set does not run correctly, you can easily identify which command is causing the problem. Once a procedure set has been tested, enter the SET REASSURANCE OFF command and then use the <set name> or EXECUTE <set name> to suppress the display of the commands as they are entered.

The SET long command uses variables and dimension labels rather than names.

4.14.3 Saving a Procedure

Procedure

1. Choose Exit from the Procedure menu. The Confirm dialog box appears, prompting you to save your changes.
2. Click Yes to save your changes. The Application Server command window appears.

Other options include:

SAVE Saves the set and remains in the editor.

QUIT Returns to the Application Server command window without saving changes.

Prints the contents of the set.

PRINT

GO Saves the changes and executes the commands.

TIDY Indents and capitalizes INPUT and OUTPUT statements for easy reading.

EXIT Requests confirmation to save and compiles the set. See Note below.

Notes:

If you choose quit instead of Exit, your changes are not saved and your procedure is not compiled.

If you choose File Exit to quit the editor, the editor attempts to compile the set. However, certain procedures, such as those that contain control variables, or enter an Application Server subsystem, cannot be compiled. If you attempt to compile such procedures, or choose File Exit, you will receive an error; choose File Save and then File Abandon Changes instead.

4.14.4 Saving Selected Information and Layout to a Procedure

Procedure to save selected information and layout to a procedure

Enter the SAVE STATUS command.

A procedure set is created with the appropriate select statements and across/down layout.

Procedure to review the commands stored in the procedure

Use the TYPE command in a procedure, for example:

SAVE STATUS salesproc SYMBOLIC

TYPE PROCEDURE salesproc

Select VARIABLES SALES

Select Dimension MERCHANDISE, FOOTWEAR, CLOTHES, SIMPSON

Select Dimension TYPE ACTUAL, BUDGET

Across VARIABLES, TYPE Down MERCHANDISE, TIME

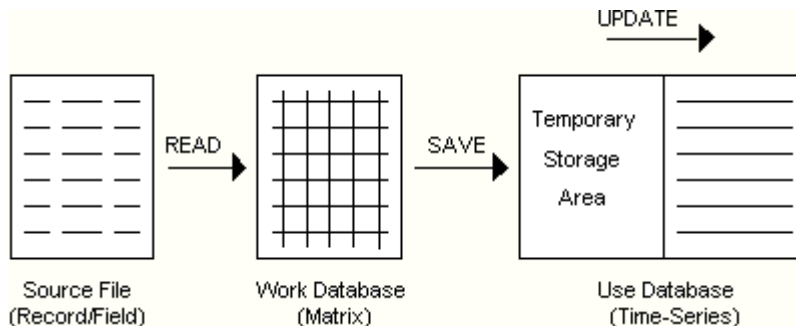
Set Long

4.15 Importing data

Internally, importing data can be viewed as a three-stage process:

Procedure

1. Data from the source file is read into the Work database. As shown in the diagram, data in the source file is in record/field or row/column format, and data in the Work database is in a matrix format.



2. Data from the Work database is converted from the matrix format to the Application Server format, and saved in a temporary storage area in your Use database. Data is saved in the temporary storage area every 300 records, unless you specify another interval using READ SAVE integer.

If you save data in the temporary storage area in your Use database, you need to have a Work database large enough to hold 300 records, or whatever integer you specify.

3. After all the data is converted and copied to the temporary storage area, it is permanently updated in your Use database.

The temporary storage area allows data to be rolled back to the record you specify if there is a system failure. For example, if you specify 600 and there is a system failure while reading record 650, data is rolled back to record 600. You can restart the import process at record 601, instead of at record 1.

If you specify READ UPDATE SAVE integer, data is permanently updated in your Use database at the same rate it is originally read from the source file.

4.16 Frequently Asked Questions

4.16.1 How can I estimate my database size when it contains data for all time periods?

Since the biggest portion of an Application Server database is the time series data for measures, the best way to estimate database size is to create all your dimensions and load and consolidate one measure for one time period. This will enable you to estimate the number of actual combinations for your measures.

Note: If all of your measures are not dimensioned by the same set of dimensions, you will need to consolidate one for each combination of dimensions.

The number of actual combinations for the consolidated measure can be determined by issuing the Rollup Editor command SHOW COUNT. The number that displays in parentheses at the end of the heading line is the number of actual combinations.

Once you know this number you can calculate the space required to store data for all 8 byte measures with the following formula:

$$((8 \text{ bytes} * \text{number of periods}) + 32) * \text{actual combinations} * \text{number of 8 byte variables}$$

The formula to calculate the space required to store data for a 4 byte measure is:

$((4 \text{ bytes} * \text{number of periods}) + 32) * \text{actual combinations} * \text{number of 4 byte variables}$

Add these two numbers together to get a rough estimate of database size. Additional space will be required for the sets in your database (dimensions, procedures, reports, and so on) and for attribute variables, but this is small in comparison to the amount of space the timeseries data occupies.

4.16.2 How can I change the fiscal calendar on my database?

The SET FISCAL command can be used only once on a database. It must be issued before you create your business measures because the calendar setting affects the way the data is stored. If you have already created your measures and loaded data, then you must follow the procedure below to change the fiscal calendar:

Procedure

1. If the original data source is no longer available, select the dimension input members and use ACCESS EXTERNAL WRITE to create a text file of the non-consolidated timeseries data.
2. Delete all business measures and data on the database.
3. Issue the new SET FISCAL command.
4. Create the measures.
5. Load the data from source or from the file created using ACCESS EXTERNAL WRITE.

Note: If you have a custom calendar set with the SET FISCAL USER STARTING .. ENDING command, you can re-run the procedure that defines the calendar when measures exist on the database. In this case, do not modify the existing ENDING dates in the procedure. You should only add more period end dates to the SET FISCAL USER STARTING .. ENDING command.

4.16.3 How can I determine what to label columns in my DESCRIPTION statement?

After you set your ACROSS and DOWN, go into the ACCESS EXTERNAL sub-system and issue the SHOW command. This will list how Application Server expects the fields to be labeled. You can have Application Server create a procedure that contains the DESCRIPTION statement by issuing the SAVE command from ACCESS EXTERNAL, for example, SAVE DATA_DESC. The DATA_DESC procedure that is created may need to be modified to match your data file exactly, but the difficult part, labeling the ACROSS fields, is done.

4.16.4 How can I calculate a % of Total column in a report?

The easiest way to do this is to order your Region dimension hierarchically and have it as your last down. The result member of the Region dimension will be the first row displayed. You can use an ORDER statement in the SET-ENDSET block of your report to calculate the percentage column. A simple example is shown below:

SET

ORDER 0, 1, '% of Total' = c1 % value(row[1],1)

decimal 0, 2

ENDSET

ROWS ALL

4.16.5 How can I determine the .INI file Application Server is using?

The default .INI file used by Application Server is LSSERVER.INI. The base file name is the same as the executable. You can specify a different .INI file by using the -inifile switch. Examine the Properties of the icon you use to start Application Server. If an alternate .INI file is being used, it will be listed on the command line after LSSERVER.EXE. Application Server will look for this .INI file in the Windows directory unless a full path is specified.

4.16.6 Why won't my construct DESCRIPTION statement work for loading data?

The error message you are getting probably says something like, "There is no Corresponding Field for *dimension*." When you CONSTRUCT a dimension from an external file, the DESCRIPTION statement you use in ACCESS EXTERNAL names the fields so that they match the levels in your dimension. For example, your GEOGRAPHY dimension may have field names like CITY, STATE, REGION, COUNTRY.

When you load data, however, you must name the fields in the external file so they match your dimension names. You really only need the input field when you load data, so you should change your DESCRIPTION statement for the data load and name the CITY field GEOGRAPHY. You can leave the other fields that are part of the GEOGRAPHY dimension as they are and Application Server will ignore them.

Also, when you are constructing a dimension, you do not need a field for TIME. Your DESCRIPTION statement must account for TIME when loading data or, if the entire file represents a single time period, you must have TIME as your first DOWN dimension.

4.16.7 Where does the UNIX version put newly created databases?

The UNIX client/server version of Application Server creates files (databases, trace files, and so on) in the working directory on the server. When you connect to Application Server from a client PC, the working directory will be the HOME or login directory of the UNIX user ID you have in your .INI file. The filename will be in upper case letters. To force Application Server to create a database in a different directory, define an environment variable for that database in the client/server startup script, lsstcp.sh, before you issue the CREATE DATA command.

Data files that you want to load into Application Server should be also located in the user HOME directory. By default, Application Server expects these load files to have upper case names. If the file name is in lower case, surround the name in single quotation marks (' ') when you reference it.

4.16.8 How can I remove data for some but not all of my observations?

You can remove data before or after a point in time by using the BEFORE or AFTER keyword on the REMOVE DATA command. For example, use the command below to remove Sales data for all months after March 98:

REMOVE DATA Sales AFTER Mar 98

You cannot use the keywords BEFORE and AFTER together to remove data in the middle of a date range. The only way to do this is to calculate null or missing values into the period or periods. To

remove Sales data for March 98 when the measure contains values for January to December 98, issue the following command:

CALCULATE Sales = MISSING PER Mar 98 FULL

You can use a question mark (?) in place of the word MISSING. Be aware that this operation could take as long as a database consolidation.

4.16.9 How can I load data that has a comma for the decimal separator in numbers?

Application Server expects external data to have the decimal separator specified with the SET DEFAULT command. Using the SET VARIABLE command to change the separator on individual measures will not make a difference.

The keyword used to set the decimal separator is POINT. To load data that has a comma for the decimal separator issue the following command prior to the data load:

SET DEFAULT POINT ','

4.16.10 How can I load data that has a field for year and a field for each month?

The table below presents sample data in the format described:

REGION	PRODUCT	DATA	YEAR	JAN	FEB ...
Boston	P001	Dollars	1998	1234	3214
Boston	P002	Dollars	1998	3251	6123
Boston	P001	Cases	1998	2143	5124
...
San Diego	P1234	Cases	1999	1698	3297

Use a WHERE clause on the SELECT to retrieve data a year at a time. Also, use aliases to rename the DATA and the JAN, FEB, ... fields so they are in the format required by Application Server. For example:

SELECT REGION, PRODUCT, DATA VARIABLES, JAN JAN98, FEB FEB98, MAR MAR98, ..., DEC DEC99 from <table> WHERE YEAR = "1999"

5 Administration and Configuration

5.1 Debugging

Application Server provides several debugging commands to help you find and correct problems with the system, applications errors, bugs in Application Server, or client/server performance problems.

Click an underlined topic for more information.

[top](#)

[XRAY](#)

[Determining buffer size](#)

[Using KEY BOTH statement for a large dimension](#)

[Other Debugging Commands](#)

5.1.1 Top

Use the UNIX top command on an HP9000 system to view all the processes that are running, the loads on the machine, and machine statistics.

Note: If you use the top command during a consolidation, and CPU processing is very active, this might be because you are not using the Rollup editor and are using an old method of consolidation.

5.1.2 XRAY

Use the XRAY command to perform an integrity check on an Application Server database. If you suspect that a database is corrupt, XRAY might be able to find the location of the corruption. A version of XRAY is provided on each Application Server platform. You must run the XRAY command on a server, not from your client machine.

Syntax	Description
XRAY database	Evaluates all the basic structures within the database and reports whether the database is corrupt.
XRAY -t database	Provides a complete analysis of the database structure.
XRAY -s database	Lists information about all internal sets in an Application Server database. Used as a diagnostic tool to determine the structure and content of a model.
XRAY -f database	Reclaims lost space in an Application Server database.

Note: Running the XRAY command can take a significant amount of time, in some cases as long as a database consolidation.

5.1.3 Determining Buffers Setting

The amount of memory used by Application Server is determined by the number of buffers it creates in memory, and the size of the buffers. Application Server creates the maximum number of buffers possible in the memory available to it, up to the value set for buffers. To optimize data load and consolidate performance, you should set buffers as high as possible, but at a level that ensures they are created in real memory rather than cache or swap memory.

Available memory and buffers

The value you specify for the DEFAULTMEMORY environment variable or the SET MEMORY command determines the amount of memory available to Application Server. If the variable

DEFAULTMEMORY is not set at startup, Application Server uses all available memory to create buffers.

If Application Server runs out of real memory for creating buffers, it uses cache or swap memory, which is less efficient and should be avoided. When determining the available memory value, you should reserve enough real memory for the operating system and priority processes, while ensuring that Application Server has as much real memory available to it as possible.

The maximum number of buffers that can be created in available memory is initially determined by the Buffers value on the Application Server user account. The default value of 2,000 is usually sufficient for query users. However, you can use the SET BUFFERS command to increase buffers for a specific process, such as a data load and consolidate. The SET BUFFERS command sets buffers for the current session only — the next time the user logs in, the Buffers value on the user account is used.

Determining buffer size

The size of the buffers that Application Server creates in memory is determined by the page size of the database set that stores time series records. The page size is determined by the observations value that is set when the database is created, and by the block size of the database. A page is made up of the number of database blocks taken to store the maximum number of observations for an 8-byte variable. The calculation of page size is based on 8 bytes, even if the database has only 4-byte variables.

For example, if observations is set to 2,000, then page size must be at least 16,000 bytes (that is, 8 bytes * 2000 observations). If the block size of the database is 8K, there are two blocks per page, and the buffer size is 16K. If observations is set to 1,500 and block size is 8K, page size and buffers are again 16K. Only 12,000 bytes (that is, 8 bytes * 1500 observations) are needed to store a fully populated time series, but page size will always be made up of full blocks. If block size is set to 4K, a database with 1,500 observations will have a page size and buffer size of 12K.

Determining the number of buffers

Once you know both the buffer size and the amount of memory available to Application Server, you can divide memory by buffer size to determine the approximate number of buffers to set for optimum performance. If you have set a value for DEFAULTMEMORY, or issued a SET MEMORY command, you should ensure that you set buffers high enough to take advantage of this setting. If you set buffers too high in relation to the amount of available memory (if buffers * page size is greater than the available memory), Application Server will not be able to create all the buffers you have requested.

To determine the observations setting for the model, use the Supervisor command SHOW DATA.

To determine the DEFAULTMEMORY or SET MEMORY value, and the Buffers setting for your Application Server session, use the SHOW SETTINGS command.

The importance of determining the correct buffer size

Although it is important to have sufficient memory to perform an operation, there can be a cost if you set your buffers too high.

When you issue the SET BUFFERS command, as well as allocating memory for buffers, Application Server allocates a table in contiguous memory with a 54-byte entry for every buffer. For example, if you set 2,000 buffers, the table contains 2,000 entries and occupies 108,000 bytes of memory.

Each allocated buffer that you do not actually need prevents 54 bytes of memory from being used by other applications. For example, if you set 3,000 buffers but need only 2,000 buffers, you allocate 54,000 bytes of memory to Application Server that it does not require and which cannot be used by other applications. For most computers, this relatively small amount of memory does not cause a problem, but it can become an issue if you set buffers to be much higher than actually

required. Therefore, you should accurately determinate the buffer size your database actually requires.

5.1.4 Using KEY BOTH Statement for a Large Dimension

You should construct a large dimension with a KEY BOTH statement in order to build an index for the dimension. This option optimizes the process of selecting members from the dimension.

5.1.5 Other Debugging Commands

RETURN, ON ERROR	You can add the RETURN command to a procedure for debugging. The procedure stops when it reaches RETURN. You can use ON ERROR and then execute a RETURN so that when Application Server encounters an error, a specific procedure can be run.
SHOW EXACTLY WHERE	Searches a set and lists every line in a report, procedures, and logic that uses the specified word. When you rename a variable, all occurrences are automatically changed.
BEGIN, END, &	You can use BEGIN and END to block off continuing commands and avoid the use of an ampersand (&).
UNIX Core File	If an Application Server process running on a UNIX machine does not complete successfully, a core file may be created in your UNIX account's home directory. You can determine if any processes are still running by issuing the following command: ps -ef grep lsstcp The following command stops leftover processes on most UNIX machines: kill "process_id"

Uninterrupted UNIX Batch

If you are running in a client/server mode with a UNIX server and a client workstation connected via telnet, you may want to run a very long job, such as loading or consolidating data.

Generally this type of job is run when you are not logged in because the link between your client workstation and the UNIX server is broken, the job stops running. To avoid this, use the UNIX nohup command, for example:

nohup batchlss.ksh &

In this command the ampersand (&) runs the batchlss.ksh script as a background process, and nohup ensures that batchlss.ksh is protected from hangups and terminations. Two example scripts for running a batch process are installed with Application Server. These are batchlss.ksh, for use with the Korn or Bourne shell, and batchlss.csh, for use with the C shell.

These scripts define all the required environment variables for running Application Server and then execute Application Server with the following commands:

exec \$LSSHOME/lss <batch1 >batch1.out 2>batch1.err

in batchlss.ksh, or:

\$LSSHOME/lss <batch1 >batch1.out 2>batch1.err

in batchlss.csh. These statements define the file batch1 as the source of all input that a user would normally enter from a terminal. Output is written to the file batch1.out and errors are written to the file batch1.err. You can create multiple copies of the batch scripts to run different jobs.

UNIX commands that monitor the trace file

<code>cat tracefile</code>	Lists the trace file for as far as the job has run.
<code>tail tracefile</code>	Shows the last ten lines of the trace file.

Maximums and Limits

cron	This is an automatic scheduler. It sends mail to the user ID if something goes wrong.
Small and Large Keys	<p>If you are using a UNIX version of Application Server, you can create an external snapshot file and use the grep utility to search the file for every occurrence of a negative key. For example:</p> <pre>cat snapshotfile grep key</pre> <p>The output from this will show you whether you have a negative key that identifies a small key, large key problem. If you have this problem, remove all consolidated data from your database, dump the database to an external file, and load the dump into a new database.</p>
How MULTIPLE works	When you create a database, Application Server sets the default multiple value to six. This means that when you create a time series, space is set aside for six points of data. As these are filled, the database does not change in size. When a seventh point is added the database will almost double in size as space is set aside for an additional six points of data. The number of points space is set aside for can be controlled by using the MULTIPLE <i>n</i> keyword when creating a database. Note that a MULTIPLE setting is valid only for variables that are single or double precision floating (4 or 8 bytes, nonsparse).

5.2 Maximums and Limits

5.2.1 Application Server Maximums

<u>Database maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
Partition size	1	2GB (4GB for NT)	NA
# partitions	0	32	1
Total database size	102,400 bytes	unlimited	Blocksize x # blocks
Block size	512	128K	16K
# blocks	200	Varies	200
# observations	100	32,000	100
# multiples	1	255	6
# open Application Server databases	1	130	NA
# open files (Use the LSSFILES environment variable in LS.INI to change the maximum number of open files.)	NA	OS-specific	64
# chars in a database name (unless the database is partitioned)	1	96 bytes	NA

Note: For distributed databases, the string specifying the dimension name and the remote database name (for example, product;maindb) cannot exceed (maximum number

Database protection key of characters in 96 bytes – 3).
63

<u>External source maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# fields (columns) in a DESCRIPTION command (ACCESS EXTERNAL)	1	2048	NA
# chars in a file name (full path)	NA	64	NA
Line size of a text file to read in:			
Windows	NA	64K	NA
UNIX	NA	64K	NA
Line size of other file types	Application Server accepts the limitations set by the database management system.		
Maximum number of open files (per CONFIG.SYS):			
Windows	NA	255	NA

<u>User maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# chars in a user name	1	96 bytes	NA
Password		36 bytes	
Usage	NA	NA	Exclusive
Work database blocks	50	Varies	130,000
Work database blocksize	512	128K	16K
# buffers	20	64,000	10,000
# maxlogins	1	32,000	255
User access key		48	

<u>Command line maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# chars on a command line	3	256	NA
# chars of a command to type	3	Full command	NA

<u>Dimension maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
ACROSS/DOWN command	TIME and VARIABLES	12 dimensions, TIME, and VARIABLES	NA
# dimensions that measures are dimensioned by in a view	0 dimensions TIME and VARIABLES	You cannot have more than 22 unique dimensions that the variables in	NA

 Maximums and Limits

	must be included in the view	the view are dimensioned by.	
# members in a dimension	1	unlimited	NA
# levels within a dimension	1	256	NA
# classes within a dimension	0	64K	0
# hierarchies in a dimension	1	256	NA
# chars in a member name	1	96 bytes	NA
# chars in a member label	1	250 (labels are truncated at 250 bytes)	NA
# User-Defined Hierarchies per dimension	0	255	0
# members per User-Defined Hierarchy	0	100,000	0
<u>Set maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# chars in a set name	1	96 bytes	NA
# sets in a database	NA	1,000	NA
# attributes in a set	NA	255	NA
<u>Variable maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# variables	0	10,000	NA
# chars in a variable name	1	96 bytes	NA
# chars in a variable label	1	50	N/A
# chars in a control variable	1	255	NA
# bytes in a virtual variable expression	1	999	NA
# chars in a text variable value	1	255	NA
# chars displayable in a text variable value	1	50	NA
# dimensions a variable can be dimensioned by	0	12, plus Time	0
maximum integer length	NA	10 digits	
Time-series combinations		1.8446744 e19	
<u>Attribute maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# attributes per dimension	0	255	NA
# members per attribute	1	unlimited	NA

# characters in an attribute name	1	96 bytes	NA
<u>Report maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# rows on a page	NA	4K per page	NA
<u>User-Defined Hierarchy maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# User-Defined Hierarchies per dimension	0	255	NA
# members per User-Defined Hierarchy	1	unlimited	NA
<u>Miscellaneous maximums</u>	<u>Minimum</u>	<u>Maximum</u>	<u>Default</u>
# case statements per security procedure	NA	1,000	NA

5.2.2 Updating the Measure Limits

By default, you can display up to 5,000 measures. If your users want to display more than 5,000 measures, follow the steps in this section to increase the limitations on measures.

Do one of the following:

When Application Server is installed on UNIX, add MAXSETS parameter as part of the Application Server start up (for example in `lsstcp.sh`) as follows:

```
MAXSETS=x
export MAXSETS
```

When Application Server is installed on Windows, add MAXSETS to the `lsserver.ini` file as follows:

```
[Windows]
MAXSETS=x
```

Where x is a number greater than the number of measures you currently have. The default setting is 5,000.

5.2.3 Binary Object Support in Application Server Databases

Binary files of any format can be stored in Application Server databases. They are stored as sets of one of the following types:

AUDIO

BINARY

MULTIMEDIA

PORTFOLIO

VIDEO

The set type chosen need bear no connection with the contents of the set. These set types are provided for convenience only.

Creating a Dump File with Multiple Partitions

The following commands support binary set types:

ASK

COPY (including COPY to or from a LOCAL file)

DIRECTORY

EXHIBIT

RECOVER

REMOVE

RENAME

SHOW

DUMP will copy binary sets to EXTERNAL files of the same name, and insert a COPY command into the DUMP file to copy them back to the database. The DUMP command will not compress binary sets.

The following commands do not support binary set types:

Editor

PRINT

SET

TYPE

5.3 Creating a Dump File with Multiple Partitions

By default, one large dump file is created. It can exceed 2Gb up to 64-bit. If you do not want a database dump to exceed 2GB, you can split the dump across multiple files (partitions). To dump a database to multiple partitions, you specify a wildcard within the dump file name, for example DUMP??. You can also specify NOLARGEFILES to make sure that each partition is less than 2Gb. You can also specify the SIZES keyword to specify the exact sizes of each partition. Application Server replaces the wildcard characters with partition numbers, for example DUMP01, DUMP02, and so on. If you have chosen to store the data and database definition together, you should specify the wildcard within the <filename> parameter; if you have chosen to store the data and database definition separately, you should specify the wildcard within the <datafile> parameter. For example:

```
DUMP dump?? SIZE 1G
```

```
DUMP dump?? NOLARGEFILES
```

```
DUMP maindump DATA dump?? SIZE 1G
```

Location of dump files in Windows

In Windows, dump files are placed in the Home directory. However, you can specify alternative locations by editing the LSSERVER.INI file before executing the DUMP command:

```
[Windows]
```

```
dump##=pathname\dump##
```

Substitute dump## with the name of a dump file partition, and substitute pathname with the directory into which you want the dump file to be placed. You should create a separate entry for each partition, for example:

```
[Windows]
```

```
dump01=c:\db\dump01
```

```
dump02=d:\db\dump02
dump03=e:\db\dump03
```

Location of dump files in UNIX

In UNIX client/server, dump files are placed in the directory defined by the LSSHOMES environment variable. To specify a different location for the dump files, add environment variables to lssstcp.sh, for example:

```
dump##=pathname/dump##
export dump##
```

Substitute dump## with the name of a dump file partition, and substitute pathname with the directory into which you want the dump file to be placed. You should create a separate entry for each partition, for example:

```
dump01=/db1/dump01
dump02=/db2/dump02
dump03=/db2/dump03
export dump01
export dump02
export dump03
```

5.4 Configuring Application Server for International Use

When Application Server is installed, it detects the language setting of your system and installs the appropriate language files.

By default, Application Server uses the Regional Settings from the Windows Control Panel to define the language, decimal separator, thousands separators, and number of decimal places.

Overriding the date settings

The SET DATE command controls the way Application Server formats dates. The Regional Settings in the Windows Control Panel has no effect on Application Server date formatting.

Overriding the currency symbol

The SET DEFAULT CURRENCY defines the currency symbol used by Application Server. In addition, you can specify a currency symbol for a specific variable with the SET VARIABLE CURRENCY command. The Regional Settings in the Windows Control Panel has no effect on the currency symbol used.

Overriding the default settings for thousands separator, decimal separator, and number of decimal places

The decimal separator, thousands separator, and number of decimal places are defined with the SET DEFAULT command. For example, you could enter SET DEFAULT DECIMALS 3 to display values with three decimal places. To revert to the Regional Settings in the Windows Control Panel, use the LOCALE keyword, for example SET DEFAULT DECIMALS LOCALE.

Overriding the thousands separator, decimal separator, and number of decimal places for a specific variable

You can override thousands separator, decimal separator, and number of decimal places for a specific variable by using the SET VARIABLE command. For example, you could enter SET VARIABLE Sales DECIMALS 0 to display Sales values with no decimal places. To revert to the Regional Settings in the Windows Control Panel, use the LOCALE keyword, for example SET VARIABLE Sales DECIMALS LOCALE.

ABS()

6 Application Server Command Reference

6.1 ABS()

Use

ABS returns the absolute value of a variable or expression.

Syntax

```
CALCULATE <result> = ABS(<input>)
```

Parameter	Description
<result>	Name of the result variable. You must enclose names with embedded spaces and names that contain special characters with single quotation marks (' ').
<input>	Name of the variable or arithmetic expression whose absolute value you want calculated.

Example: ABS

...This example returns a value of 21:

```
CALCULATE a = ABS(7*(-3))
```

6.2 ACCESS DBASE

Use

ACCESS DBASE starts the Access subsystem, enabling Application Server to read or write data to dBASEIII and IV files with a .DBF extension.

Notes:

The field name in the database field must match either the name of the dimension being read in with the READ or the name of the dimension level used in the CONSTRUCT command.

All dimension information must be part of the file you are accessing.

ACCESS DBASE does not support fields defined as float in DBF (dBASE). The data must be held in a field defined as numeric in the .DBF file. Use ACCESS LSLINK for this data type.

Syntax

```
ACCESS DBASE
```

Related Application Server commands

BEGIN-END

CONSTRUCT

CREATE

DEFINE SYNONYM

END

PEEK

PROCEDURE

READ
RECONSTRUCT
USE
WRITE

Example

...This example reads in data from DRINKS.DBF. (Do not specify the .DBF extension. When you use ... ACCESS DBASE, Application Server automatically looks for .DBF files.) Application Server uses ...this data to create the dimensions Product and Channel, and the variables Actsales and Budsales.

```
ACCESS DBASE
USE drinks
CONSTRUCT product LEVEL product
COMPILE dim product
CONSTRUCT channel LEVEL channel
COMPILE dim channel
END
CREATE VARIABLE actsales BY product, channel
CREATE VARIABLE budsales BY product, channel
SELECT VARIABLE actsales, budsales
SELECT PRODUCT INPUT
SELECT CHANNEL INPUT
SET PERIOD Jan 95 - Jan 96
ACROSS variables DOWN product, channel, time
ACCESS dbase
USE drinks
...This line sets the field to match the dimension name
CREATE time=date
READ
END
```

6.3 ACCESS EXTERNAL

Use

ACCESS EXTERNAL starts the Access subsystem, enabling Application Server to read or write data to text or binary files. It is possible to read and write files that exceed 2Gb.

Note: To read from or write to external text or binary files, you have to specify the external file format using DESCRIPTION. Also, all dimension information must be part of the file you are accessing.

Note: If you know that the external data is all uppercase and has no internal spaces in any dimension fields, then use the UPPERCASE and NOSPACE keywords on the *type* field of DESCRIPTION in ACCESS EXTERNAL to save CPU.

ACCESS EXTERNAL**Syntax**

ACCESS EXTERNAL

Keywords

BEGIN-END
CONSTRUCT
CREATE
DESCRIPTION
END
PEEK
PROCEDURE
READ
RECONSTRUCT
SHOW
USE
WRITE

Example 1**ACCESS External****USE 'dim2.txt'****BEGIN**

DESCRIPTION fixed
'SKU2' text 10 1
'Sku_desc' text 15 17
'Busprod' text 3 49
'Busprod_desc' text 30 54
'Prodfamily' text 3 74
'Prodfam_Desc' text 30 79
'Subfamily' text 3 109
'Subfam_Desc' text 30 114

END**PEEK****CREATE Business = substr(Busprod,1,1)****CREATE Sku = Sku2 + Sku_desc****CONSTRUCT SKUSRP level SKU2,Subfamily,Prodfamily,busprod,business labels****?,Subfam_desc,Prodfam_desc,busprod_desc,? repla calcu****END****Example 2**

...This example procedure produces a file called TEXT:

SELECT PRODUCT**SELECT REGION #1****SELECT SALES****SET PERIOD 1/99**

ACROSS VARIABLE,TIME DOWN PRODUCT,REGION

ACCESS EXTERNAL

USE TEXT

BEGIN

DESCRIPTION FIXED

PRODUCT TEXT 20

REGION TEXT 20

'SALES, Jan 99' NUMERIC 10.2

END

WRITE OVERWRITE

END

...The output of the file TEXT would look like this:

Command Center	North East	289.00
LightShip	North East	285.00
TimeBase	North East	359.00
FCS	North East	5.00
Navigator	North East	833.00
Total Product	North East	1771.00

...When you are in ACCESS EXTERNAL, you might want to write data out and control the decimal point

... in one of two ways. To print a number with an explicit decimal point, as in: 29.54

...For example, to get a field ten characters wide with 2 decimal places and an explicit decimal point.

SALES NUMERIC 10.2

...To print that number out with two implied decimal places, as in: 2954

SALES NUMERIC 10(2)

...The parentheses indicate "implied" and must be present.

...When reading data, the same conventions apply. To read "2954" with two implied decimals, enter

SALES NUMERIC 10(2)

6.4 ACCESS LSLINK

Use

ACCESS LSLINK starts the Access subsystem with an access link to data stored in relational databases on several platforms. ACCESS LSLINK is available upon installation and configuration of Link. You can access more than 15 data sources for data retrieval.

Note: All dimension information must be part of the file you are accessing.

Syntax

ACCESS LSLINK

Keywords

BEGIN-END

CONSTRUCT

CREATE (Access) or TS CREATE

ACCUM()

DEFINE SYNONYM
END [LEAVE]
PEEK
PROCEDURE
READ
RECONSTRUCT
WRITE

Example 1

...Use ACCESS LSLINK to connect to a dBASE file, and then constructs and compiles a dimension.
... The first SELECT command selects the desired fields in the source file, and PEEK displays the first
... ten records. After you enter PEEK, enter SELECT again, because PEEK removes the selection.
...Use TS CREATE statements when the name of the external field in the .DBF file is different from the
.. Application Server internal name. Application Server constructs .and then compiles the dimension.

ACCESS LSLINK

CONNECT dbase

SELECT city, state_code, state, area from cities.dbf,states.dbf where cities.state_code = state.code

PEEK ONLY 10

SELECT city, state_code, state, area from cities.dbf,states.dbf where cities.state_code = state.code

TS CREATE stcode=state_code

TS CREATE region=area

PEEK CREATE ONLY 10

SELECT city, state_code, state, area from cities.dbf,states.dbf where cities.state_code = state.code

CONSTRUCT location LEVEL city, state, region LABEL ?,stcode,?

END

COMPILE DIMENSION location

Example 2

...Read in a memo text file from a dBASE file using this format for the SQL SELECT statement:

SELECT dimname, SUBSTR(textmemofield,1,200) textvariable FROM database.dbf

...Note: An Application Server text variable only holds 255 characters, so that is all

...that you can read from the dBASE memo text field.

6.5 ACCUM()

Use

ACCUM adds values within a specified time period.

Syntax

CALCULATE <result> = ACCUM(<variable>, <periodicity>)

Parameter	Description
<result>	Name of the result variable. Can be a temporary variable.
<variable>	Name of the variable whose values you want to accumulate. You must enclose names with embedded spaces and names that contain special characters with single quotation marks (' ').

<periodicity> Accumulates values at one of the following frequencies: yearly, quarterly, monthly, lunar, bimonthly, weekly, biweekly, daily, or hourly.

Notes:

ACCUM only works with the periodicities listed above, as these work on the SET PERIOD range. Periodicities that work on SET LATEST (such as rolling periodicities) do not work with ACCUM.

Do not use Accum() or any other time function in the definition of a virtual variable. An error would result if you display that virtual variable in the DOWN and a timeset in the ACROSS because there is no periodicity until a LIST or DISPLAY is executed.

Example

...Consider this statement:

CALCULATE Totalsales = ACCUM(Sales, quarterly)

...The following quarterly values yield Totalsales as shown:

	1990				2000			
Sales	13	10	9	15	14	10	—	—
Totalsales	13	23	32	47	14	24	—	—

6.6 ACROSS/DOWN

Use

Use ACROSS/DOWN to define how data is viewed, imported, or exported. The ACROSS/DOWN command is updated to include attributes in a view.

Syntax

```
ACROSS {[<dimensions> ] [ <attributes> ] | NOTHING} DOWN
      {[<dimensions> ] [<attributes> ] | NOTHING}
      {SWITCH}
```

Parameter	Description
ACROSS	Displays the specified dimensions and attributes across the top of each page.
DOWN	Displays the specified dimensions and attributes down the side of each page. The last down dimension or attribute appears as a line item on a page. All other down dimensions are page breaks.
<dimensions>	One or more dimension names, separated by commas.
<attributes>	One or more attribute names, separated by commas.
SWITCH	Switches the across and down view.
NOTHING	Lists all dimensions, attributes, and variables in one direction, skipping the direction that contains the NOTHING keyword. Dimensions, attributes, and variables display either all across the page or all down the page. Use NOTHING when you are reading in data from an external file formatted so that each column represents one dimension and the data exists in one data column (third normal form). You must name the data column in the external file VALUE. The ACCESS EXTERNAL DESCRIPTION statement must specify the VALUE field.

Example 1

...This example displays combinations of the dimensions Region and Time as columns across the top
... of each page, and rows of Variables for each Product down the side:

SELECT Region east, west

SELECT cogs, sales

ACROSS/DOWN

SELECT Product 'Processor Chips', 'Color Monitors'

ACROSS Region, Time **DOWN** Product, Variables

LIST yearly

...The output of this command would be:

	East	West
	1999	2000
PRODUCT Processor Chip		
Cost of Sales		
Sales		
PRODUCT Color Monitors		
Cost of Sales		
Sales		

Example 2

...This **ACROSS NOTHING DOWN** command displays all dimensions and variables down the page:

SELECT Product 'Processor Chips'

SELECT Region East, West

SELECT cogs, sales

SELECT Type Actual

ACROSS NOTHING DOWN Product, Region, Variables

LIST yearly **PERIOD** 2000

...The output of this command would be:

PRODUCT Processor Chips

REGION East

2000

Cost of Sales 92,702,386

Sales 129,452,856

PRODUCT Processor Chips

REGION West

2000

Cost of Sales 123,073,628

Sales 172,020,562

...This **ACROSS NOTHING DOWN** command is used to read data from an external file that is

...formatted similarly to the one below, where each column represents a dimension and the data is

...contained in a data column:

Region	Product	Time	Variable	Value
Boston	80386	Jan 00	SALES	200
New York	80386	Jan 00	SALES	200 ...

...The data column in the external file must be named **VALUE** and the **ACCESS EXTERNAL**

...**DESCRIPTION** statement must specify the **VALUE** field:

SELECT Product

```

SELECT Region
SELECT VARIABLES
SET PERIOD 2000
ACROSS NOTHING DOWN Product, Region, Variables, Time
ACCESS EXTERNAL
USE sourcefile
BEGIN
    DESCRIPTION statement
END
READ
END

```

Example 3

...This example switches the ACROSS and DOWN display from ACROSS Region, Time DOWN

...Product, Variables, to ACROSS PRODUCT, Variables DOWN Region, Time:

ACROSS SWITCH

LIST yearly

...The output of this command would be:

```

                Processor Chip Color Monitors
          Cost of Sales   Sales   Cost of Sales   Sales
REGION East
    1999
    2000
REGION West
    1999
    2000

```

6.7 ACRSLIH()

Use

ACRSLIH returns the depreciation of low-income housing.

Syntax

CALCULATE <depreciation> = ACRSLIH(<asset>, <term>, <start>, <tax_month>, <method>)

Parameter	Description
<depreciation>	Name of the result variable with the calculated time-series depreciation.
<asset>	Asset scalar value.
<term>	Number of years of asset life.
<start>	Asset start date.
<tax_month>	Month of the year (1 to 12) in which the tax year begins.
<method>	Depreciation method, where: 0 = ACRS 1 = Straight line

ACRSPP()

6.8 ACRSPP()

Use

ACRSPP returns the depreciation of personal property.

Syntax

CALCULATE <depreciation> = ACRSPP(<asset>, <term>, <start>)

Parameter	Description
<depreciation>	Name of the result variable with the calculated time-series depreciation.
<asset>	Asset scalar value.
<term>	Number of years of asset life.
<start>	Asset start date.

6.9 ACRSRP()

Use

ACRSRP returns the depreciation of real property.

Syntax

CALCULATE <depreciation> = ACRSRP(<asset>, <term>, <start>, <tax_month>, <method>, <domestic>)

Parameter	Description
<depreciation>	Name of the result variable with the calculated time-series depreciation.
<asset>	Asset scalar value.
<term>	Number of years of asset life.
<start>	Asset start date.
<tax_month>	Month of the year (1 to 12) in which the tax year begins.
<method>	Depreciation method, where: 0 = ACRS 1 = Straight line
<domestic>	Location of the property, where: 0 = U.S. 1 = Outside the U.S.

6.10 ADD ACCESS (Supervisor)

Use

A Supervisor command that adds an access key to a user record in MASTERDB. This enables the specified user to access all databases with the same access key.

Syntax

ADD ACCESS <key> USER <user>

Parameter	Description
ACCESS <key>	Specifies the access key to add. The <key> parameter can contain up to 16 alphanumeric characters.
USER <user>	Specifies the user ID to which the access key is added. The <user> parameter must begin with an alphabetic character; the remaining 23 available characters can be alphanumeric.

Remarks

The access mode available to the user (either READ or UPDATE) is defined in the database.

Example: ADD ACCESS

...This example adds a read access key to the Juice database and to the user named Test1.

...Create the Test1 user.

SUPERVISOR CREATE USER Test1 USEDDB Juice

...Add the access key to the Juice database.

SUPERVISOR CHANGE DATABASE Juice READ TestKey

...Add the access key to Test1 so the Test1 user can read from the Juice database.

SUPERVISOR ADD ACCESS TestKey USER Test1

6.11 ADD DATABASE (Supervisor)

Use

A Supervisor command that adds a record for a database to MASTERDB. The database might be one created on another machine, or one you defined in another MASTERDB.

Syntax

ADD DATABASE <database> [MAXACCESS UPDATE | READ] [PROTECTION <key>]

Parameter	Description
DATABASE <database>	Creates a database record in MASTERDB for an existing Application Server database (and its partitions). This might be a database created on another system, or a database you defined in a different MASTERDB. The <database> parameter specifies the name of the database to add.
MAXACCESS UPDATE READ	Specifies the maximum level of access permitted for the database. You can specify UPDATE or READ access.
PROTECTION <key>	Indicates the database had a protection key in the system where it was created and adds the specified protection key to the new database record in MASTERDB, where <key> is the name of the protection key. This does not add the protection key to MASTERDB itself.

Remarks

You should copy the Application Server database (and any partitions) to your system before you use ADD DATABASE. Otherwise, Application Server accepts the database name but marks the database as offline. The database must have the same file name on your system as the name specified in the ADD DATABASE command.

You only need to use ADD DATABASE to add the main database file; any partitions are added automatically.

Note: If you are trying to load a dump of MASTERDB that was dumped prior to version 9.3 of Application Server and contains information about partitioned APPLICATION SERVER databases, the dump file will be unloadable. You must edit the dump file to remove the PARTITION and NAME clauses from the ADD DATABASE commands, and then you will be able to load the pre-9.3 version of MASTERDB. For help doing this contact Customer Support.

Example

This example adds the Juice database to MASTERDB.

SUPERVISOR ADD DATABASE Juice

6.12 ADD USER (Supervisor)

Use

A Supervisor command that adds a one or more users to a User-Defined Hierarchy to restrict their view of dimensions.

Syntax

ADD USER <user> TO <user_defined_hierarchy>

Parameter	Description
USER <user>	One or more user names, separated by a comma, that you want to add to the User-Defined Hierarchy. The user names you specify must already exist in MASTERDB. An asterisk (*) adds all users to the specified User-Defined Hierarchy.
TO <user_defined_hierarchy>	Specifies the name of the User-Defined Hierarchy to which you want to add users.

Remarks

Once you add users to a User-Defined Hierarchy, you can reference that User-Defined Hierarchy in an INDEX USER-CASE-ENDINDEX statement in a Security procedure. In this procedure, you specify the restricted dimension view for this User-Defined Hierarchy of users.

Example: ADD USER

...Create a user named Test1 and Test2, and adds them user to a User-Defined Hierarchy named

... Test User-Defined Hierarchy. Create the users and add them to the User-Defined Hierarchy.

SUPERVISOR CREATE USER Test1

SUPERVISOR CREATE USER Test2

SUPERVISOR CREATE GROUP Test

SUPERVISOR ADD USER Test1, Test2 TO Test

6.13 ALL-ENDALL

Use

ALL-ENDALL performs calculations at the output levels of dimensions.

Note: CALCULATE FULL has the same effect as ALL-ENDALL.

Syntax

```
ALL
.
.   <logic statements>
.
ENDALL
```

Example

... Calculates profit for all input member combinations. Values for Profit are consolidated

... (rather than calculated) for output member combinations. Then, because of the ALL-ENDALL block

..., Application Server calculates the Profit Ratio rather than consolidates for all member combinations.

Profit = Sales - Expenses

ALL

'Profit Ratio' = Profit/Sales

ENDALL

6.14 ALLOCATE (Dimension or CONSTRUCT)

Use

ALLOCATE is a Dimension command and a keyword in the CONSTRUCT command that you use to define the maximum number of input, output, and result members in a dimension. Having an ALLOCATE statement in your dimension may prevent the need for Application Server to reorganize the database if you later add members to the dimension.

Syntax

ALLOCATE [UPDATE] <input>, <output>, <result>

Parameter	Description
UPDATE	Indicates that Application Server can automatically add input members to the dimension when data is loaded.
<input>	Maximum number of input members.
<output>	Maximum number of output members.
<result>	Must be 1 or 0.

Using ALLOCATE with User-Defined Hierarchies

User-Defined Hierarchies come out of the total number of outputs in the ALLOCATE statement. You can change the number of User-Defined Hierarchies up or down as long as you remain within the limits of the ALLOCATE statement and the number of static outputs.

The User-Defined Hierarchies are part of the number of outputs in the ALLOCATE statement. For example, if you want to have 200 outputs and 50 User-Defined Hierarchies, you would enter:

CUSTOM 50

ALLOCATE ...,250,...

Example 1

...The dimension Product is defined with a maximum of 100 inputs, 10 outputs, and 1 result:

ALLOCATE 100, 10, 1

Example 2

...Inputs, outputs, or results can be specified if they are known. For example, the result of the previous ... example is Total_Product, and two known products are Cars and Trucks. You enter this as:

ALLOCATE 100, 10, 1

INPUT Cars, Trucks

RESULT Total_Product

6.15 ANALYZE

Use

ANALYZE displays statistics for members of the last down dimension, based on values represented by the across dimension. ANALYZE uses the data specified by previous SELECT and

ANALYZE: Analytics Options

ACROSS/DOWN statements. Application Server groups output according to the down list and analyze according to the across list.

Syntax

ANALYZE [<option>, ..., <option>] [NOHEADING] [TABOUTPUT] [<periodicity>] [<period>]

Parameter	Description
<option>	Can be any of the following: <ul style="list-style-type: none"> Analytics options Statistical options Forecasting options Smoothing options Regression option
NOHEADING	The output has no headings.
TABOUTPUT	Tabs delimit columns.
<periodicity>	Analyzes data at one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, or hourly.
<period>	Used to specify one of the following periods: <ul style="list-style-type: none"> yty - The latest period back to the equivalent period one year ago. yrago - The latest period one year ago. current - The period set with a SET LATEST command. previous - The previous period. latest - The period set with a SET LATEST command. FITTED - Option that you can apply to the Forecasting and Smoothing options.

Note: The ANALYZE command uses the default options CASES, DEVIATION, MAXIMUM, MEAN, and MINIMUM if you do not specify any options.

Example: ANALYZE

...Calculate the total, minimum, maximum over time of each product for variables Price, Units, Revenue

ACROSS Time DOWN Product, Variables

SELECT Price, Units, Revenue

ANALYZE TOTALS, MINIMUM, MAXIMUM

6.16 ANALYZE: Analytics Options

Use

The ANALYZE command with the Analytics options displays statistics for members of the last down dimension, based on values represented by the across dimension. The ANALYZE command with the Analytics options uses the data specified by previous SELECT and ACROSS/DOWN statements. Output is grouped according to the down list and analyzed according to the across list.

Syntax

ANALYZE [<option>, ..., <option>] [<format_option>] [<periodicity>]

Parameter	Description
AUTOCORRELATION	A study of coefficients that helps determine any pattern in a time series.
CORRELATE	Linear correlation matrix for the values of a series of variables. Correlation analysis provides a measure of the degree to which variables are linearly related,

	and an index designed to give an immediate picture of how closely two variables move together.
DURBIN	Durbin-Watson statistic and autocorrelation coefficient indicates whether values in the row are serially correlated.
ERROR	Standard error of the mean, assuming the variable is a sample of a large population.
KURTOSIS	Steepness or flatness of the distribution of values in the variable.
SKEWNESS	Degree of asymmetry of values in the row around the mean.
SPEARMAN	The Spearman Rank Correlation Coefficient tests for a trend or systematic increase or decrease in a time series.
THIEL	Comparison between an actual series of data and a predicted or fitted series.
TTEST	Comparison between two groups of data to determine whether the data comes from the same population, or a comparison between a set of data and a predetermined value for significant differences in the data. For example: ANALYZE TTEST [<Double constant against which to compare>] [<format_option>] [<periodicity>]
VARIANCE	Relative dispersion of all values in the variable; the coefficient of variation.

Examples: Analytics options

This example runs the ANALYZE SPEARMAN command with the Spear dimension for the time period January 1, 1998, to July 15, 1998:

SELECT spear

1 Variable Currently Selected

SET PERIOD 1jan98-15jul98

LIST

```

07 Jan 98 14 Jan 98 21 Jan 98 28 Jan 98 04 Feb 98 11 Feb 98
SPEAR 38.0000 33.0000 5.0000 31.0000 20.0000 0.0000
18 Feb 98 25 Feb 98 04 Mar 98 11 Mar 98 18 Mar 98 25 Mar 98
SPEAR 49.0000 0.0000 8.0000 2.0000 34.0000 20.0000
01 Apr 98 08 Apr 98 15 Apr 98 22 Apr 98 29 Apr 98 06 May 98
SPEAR 0.0000 3.0000 3.0000 71.0000 4.0000 0.0000
13 May 98 20 May 98 27 May 98 03 Jun 98 10 Jun 98 17 Jun 98
SPEAR 0.0000 3.0000 25.0000 0.0000 22.0000 0.0000
24 Jun 98 01 Jul 98 08 Jul 98 15 Jul 98
SPEAR 2.0000 0.0000 0.0000 2.0000

```

1 Page of Data Listed

ANALYZE SPEARMAN

07 Jan 98 - 15 Jul 98

SPEAR Spearman Test

0.3342 = Rank Correlation Coefficient

1.8078 = T Statistic

1.7089 = T95%

Significant at 95% level, suggesting a downward trend.

6.17 ANALYZE: Forecasting Options

Use

The ANALYZE command with the Forecasting options extrapolates forecasts from historical data by using the least-squares method to relate the variable you want to forecast to time. The ANALYZE Forecasting options derive the equation $y = f(t)$ where f represents a function or a curve you specify. The most common curve is a straight line or exponential curves.

With the ANALYZE Forecasting options you can fit up to seven different curves to historical data to produce a forecast: GEOMETRIC, HYPERBOLIC, LINEAR EXPONENTIAL, MODHYPERBOLIC, RATIONAL, and QUADRATIC. You can then match your data to a curve, try more than one curve, and compare the goodness-of-fit. Include the FITTED option in your ANALYZE command to use the goodness-of-fit function.

The EXPONENTIAL, GEOMETRIC, HYPERBOLIC, MODHYPERBOLIC, and RATIONAL curves are transformed to a linear form prior to the least-squares calculation.

Syntax

ANALYZE [<option>, ..., <option>] [<periods_to_forecast>] [FITTED] [<format_option>] [<periodicity>]
[<period>]

Parameter	Description
Global options	You can use these with the Forecasting options: NOHEADING, TABOUTPUT, <periodicity>, and <period>.
FITTED	Displays the generated data by fitting the historical data to the chosen curve. The default is not to display the fitted data. The default for the <periods to forecast> option is 5.
EXPONENTIAL	Transformation of curves to a linear form.
GEOMETRIC	Forecasts based on historical data fit to a geometric curve.
HYPERBOLIC	Forecasts based on historical data fit to a hyperbolic curve.
LINEAR	Forecasts based on historical data fit to a linear curve. The STRAIGHT LINE curve $y = a + bx$ is the curve normally used for forecasting.
MODHYPERBOLIC	Forecasts based on historical data fit to a modhyperbolic curve.
QUADRATIC	Least-squares principle applied to quadratic curves that cannot be transformed directly into a linear equivalent.
RATIONAL	Rational curve fit to historical data.

Example

This example shows the information displayed when you run the ANALYZE LINEAR command with the Linear dimension for the time frame January 2000 to October 2000:

SELECT linear

1 Variable Currently Selected

SET PERIOD jan00-oct00

LIST

Jan 00

LINEAR 0.2030

1 Page of Data Listed

ANALYZE LINEAR

Jan 00 - Oct 00

LINEAR Linear Forecast

Observation	Forecast
1	0.2784
2	0.2856
3	0.2927
4	0.2999
5	0.3071

6.18 ANALYZE: Regression Option

Use

The ANALYZE command with the Regression option is a statistical method that examines the relationships between one or more independent variables, x , and a dependent variable, y . Regression determines the slope and intercept for the best line that passes through the variables x and y . The most common form of regression is multiple linear regression. You can also use the Regression option as a forecasting technique. Regression uses the least-squares principle.

Syntax

ANALYZE REGRESSION [<format_option>] [<periodicity>] [<period>]

Parameter	Description
Global options	You can use these options with the regression options: NOHEADING, TABOUTPUT, <periodicity>, and <period>.

Example

This ANALYZE REGRESSION example analyzes the regression of the fixed cost, variable cost, number of machines, and R. M. Grade dimensions for the time frame January 1997, to May 1998:

SET PERIOD jan99-may00

SELECT fixed_cost, variable_cost, no_machines, r_m_grade

4 Variables Currently Selected

SET default decimal 4

LIST

	Jan 99	Feb 99	Mar 99	Apr 99	May 99
FIXED COST	3,067.0000	2,828.0000	2,891.0000	2,994.0000	3,082.0000
VARIABLE COST	7,107.0000	6,373.0000	6,796.0000	9,208.0000	14,792.0000
NO MACHINES	21.0000	22.0000	22.0000	20.0000	25.0000
R M GRADE	129.0000	141.0000	153.0000	166.0000	193.0000
	Jun 99	Jul 99	Aug 99	Sep 99	Oct 99
FIXED COST	2,898.0000	3,502.0000	3,060.0000	3,211.0000	3,286.0000
VARIABLE COST	14,562.0000	11,964.0000	13,526.0000	12,656.0000	14,119.0000
NO MACHINES	23.0000	20.0000	23.0000	20.0000	20.0000

ANALYZE: Regression Option

R M GRADE	189.0000	175.0000	186.0000	198.0000	187.0000
	Nov 99	Dec 99	Jan 00	Feb 00	Mar 008
FIXED COST	3,542.0000	3,125.0000	3,022.0000	2,922.0000	3,950.0000
VARIABLE COST	16,691.0000	14,571.0000	13,619.0000	14,575.0000	14,565.0000
NO MACHINES	22.0000	19.0000	22.0000	22.0000	21.0000
R M GRADE	195.0000	206.0000	198.0000	192.0000	191.0000
	Apr 00	May 00			
FIXED COST	4,488.0000	3,259.0000			
VARIABLE COST	18,573.0000	15,618.0000			
NO MACHINES	21.0000	22.0000			
R M GRADE	200.0000	200.0000			

2 Pages of Data Listed

ANALYZE

Jan 99 - May 00						
	Cases	Minimum	Maximum	Mean	Std Dev	Total
FIXED_COST	17	2,828.0000	4,488.0000	3,242.7647	428.867044	55,127.0000
VARIABLE_COST	17	6,373.0000	18,573.0000	12,900.8824	3,526.992414	219,315.0000
NO_MACHINES	17	19.0000	25.0000	21.4706	1.462773	365.0000
R_M_GRADE	17	129.0000	206.0000	182.2941	22.264784	3,099.0000

ANALYZE REGRESSIONGoodness of Fit (Adjusted R²) 0.5509

Significant at 99% level

Multiple correlation coefficient(R) 0.7969
 Coefficient of determination (R²) 0.6351
 Standard error of estimate 287.3963
 F - Value 7.5430

X-Variable	T-Value	StdDev of Regression Coeff
1	3.9321	0.0494
2	-3.1992	37.2083

3 -2.7715 7.4448

F-Value 99% (tables) 5.6988

F-Value 95% (tables) 3.4063

T-Value 95% (tables) 2.1597

Analysis of Variance

Source w.r.t.	Degrees of	Sum of	Mean
Regression	Freedom	squares	square
Due to	3	1,869.07469K	623,024.8950
Deviat. from	13	1,073.75637K	82,596.6441
Total	16	2,942.83106K	

Table of Residuals

Observation	Y-Value	Y-Estimate	Residual	%
1	3,067.0000	3,272.8885	-205.8885	-6.2907
2	2,828.0000	2,763.6594	64.3406	2.3281
3	2,891.0000	2,598.2295	292.7705	11.2681
4	2,994.0000	3,036.6281	-42.6281	-1.4038
5	3,082.0000	2,969.1158	112.8842	3.8019
6	2,898.0000	3,245.0400	-347.0400	-10.6945
7	3,502.0000	3,386.3179	115.6821	3.4162
8	3,060.0000	3,105.6831	-45.6831	-1.4710
9	3,211.0000	3,046.1750	164.8250	5.4109
10	3,286.0000	3,557.3538	-271.3538	-7.6280
11	3,542.0000	3,653.8621	-111.8621	-3.0615
12	3,125.0000	3,372.1572	-247.1572	-7.3294
13	3,022.0000	2,995.1811	26.8189	0.8954
14	2,922.0000	3,304.6997	-382.6997	-11.5805
15	3,950.0000	3,442.4259	507.5741	14.7447
16	4,488.0000	4,035.3346	452.6654	11.2175
17	3,259.0000	3,342.2483	-83.2483	-2.4908

Durbin Watson statistic 2.1583

6.19 ANALYZE: Smoothing Options

Use

With the Smoothing options you can smooth historical data to obtain a trend and a forecast. The techniques used—moving averages and exponential smoothing—are empirical; their usefulness is based on experience. Include the FITTED option in your ANALYZE command to use the goodness-of-fit function.

Syntax

ANALYZE [<option>, ..., <option>] [<periods_to_forecast>] [FITTED] [<format_option>] [<periodicity>] [<period>]

Parameter	Description
Global options	You can use these options with the Smoothing options: NOHEADING, TABOUTPUT, <periodicity>, and <period>.
FITTED	Displays the generated data by fitting the historical data to the smoothed data. The default is not to display the fitted data. The default for the [<periods to forecast>] option is 5.
ADAPTIVERESPONSE	A smoothing procedure similar to SINGLEEXPONENTIAL except that the smoothing constant is calculated as a function of the data and previous errors. ADAPTIVERESPONSE is better than SINGLEEXPONENTIAL for forecasting from a series without a trend because ADAPTIVERESPONSE is responsive to changes in the pattern of the data.
DOUBLEEXPONENTIAL	Performs two weighted-averaging calculations using the results from the previous calculation as input for the next calculation. Use DOUBLEEXPONENTIAL smoothing where there is a linear trend.
MOVINGAVERAGE	A linear data smoother. The average of n adjacent observations replaces each observation. Moving averages of a data series reduce the amount of variation in the data. For a time series, moving averages are often used to eliminate unwanted fluctuations, for example to smooth the data series. You can remove cyclical, seasonal, and irregular patterns and leave only the trend. A quadratic equation is calculated to fit a curve to the averaged data. Use this equation to reconstruct smoothed data at the ends of the series. Note: You must specify two parameters. For example: ANALYZE MOVINGAVERAGE 5 4 FITTED
SINGLEEXPONENTIAL	A weighted-averaging technique where the weights decrease with the age of the data. Use SINGLEEXPONENTIAL where there is no trend.
TRIPLEEXPONENTIAL	Performs three weighted-averaging calculations using the results from the previous calculation as input for the next calculation. Use TRIPLEEXPONENTIAL smoothing where there is a non-linear trend.
WINTERS	A weighted-averaging calculation that incorporates a linear trend and a seasonal factor. Use WINTERS for forecasting where seasonality and trend are suspected in the data. For example: ANALYZE WINTERS <Random Constant> <Seasonal Constant> <Trend Constant> <obs per cycle> [FITTED] [<periodicity>]

Note: Exponential smoothing is not the best forecasting method if the series is fairly smooth (that is, if there are not positive and negative deviations from the trend).

Example

This example runs the ANALYZE MOVINGAVERAGES command to forecast the variable Wage for the next five months using the last five periods to average. The command uses the Wage dimension for the time frame January 1998 to October 1998.

SELECT wage

1 Variable Currently Selected

SET PERIOD jan98-oct98**LIST**

	Jan 98	Feb 98	Mar 98	Apr 98	May 98	Jun 98
WAGE	7800.0000	7850.0000	7955.0000	7800.0000	7775.0000	7805.0000
	Jul 98	Aug 98	Sep 98	Oct 98		
WAGE	7800.0000	7855.0000	7900.0000	7855.0000		

1 Page of Data Listed

ANALYZE MOVINGAVERAGE 5 4 FITTED

Jan 98 - Oct 98

WAGE Moving Average Forecast

Observation	Fitted Value
Jan 98	7891.7857
Feb 98	7863.5000
Mar 98	7836.0000
Apr 98	7837.0000
May 98	7827.0000
Jun 98	7807.0000
Jul 98	7827.0000
Aug 98	7843.0000
Sep 98	7860.5000
Oct 98	7887.9286

Observation	Forecast
1	7922.3214
2	7963.6786
3	8012.0000
4	8067.2857

6.20 ANALYZE: Statistical Options

Use

The ANALYZE command with the Statistical options performs basic statistical functions.

Syntax

ANALYZE [<option>, ..., <option>] [<format_option>] [<periodicity>] [<period>]

ASK

Parameter	Description
Global options	You can use these with the Statistical options NOHEADING, TABOUTPUT, and periodicity.
ADEVIATION	Average deviation of the mean of the variables.
CASES	Total number of nonmissing values in the variable.
DEVIATION	Standard deviation of the variable.
MAXIMUM	Maximum value of the variable.
MEAN	Mean value of the variable.
MEDIAN	Median value of the variable.
MINIMUM	Minimum value of the variable.
MODE	Mode value of the variable.
<periodicity>	Analyzes data at one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, or hourly.
<period>	Used to specify one of the following periods: yty - The latest period back to the equivalent period one year ago. yrage - The latest period one year ago. current - The period set with a SET LATEST command. previous - The previous period. latest - The period set with a SET LATEST command.
RANGE	Difference between maximum and minimum values in the row.
TOTAL	Total of all values in the variable.

Example: Statistical options

This example runs the ANALYZE command to determine the mean and median values of the variables, the mode value of the variable, and the total of all the values in the variable. The variables selected are *divna* and *divnb* for the time frame January 1995 to December 1995.

```
SELECT divna,divnb
```

```
SET PERIOD jan99-dec99
```

```
LIST
```

```

      Jan 99      Feb 99  Mar 99  Apr 99  May 99  Jun 99
DIVNA  16.0000  14.0000  14.0000  15.0000  16.0000  17.0000
DIVNB  18.0000  14.0000  15.0000  16.0000  16.0000  16.0000
```

```
1 Page of Data Listed
```

```
ANALYZE MEAN MEDIAN MODE SKEWNESS TOTAL
```

```

                        Jan 99 - Dec 99
      Mean      Median Mode  Skewness      Total
DIVNA  15.3333  15.5000  14.0000  0.041703      92.0000
DIVNB  15.8333  16.0000  16.0000  0.244476      95.0000
```

6.21 ASK

Use

ASK creates dialog boxes that prompt users for information. ASK can also set control variables and validate user responses automatically.

Note: Because ASK commands are usually long, you should include them in a procedure enclosed in a BEGIN-END block. You can specify one or more options within a single ASK command (see Example 3).

Note: This command is provided for backward compatibility only, and is likely to be removed in a future release.

Syntax

ASK

```
[GLOBAL <global_options> [<global_options>]*]
[TITLE '<text>']
[ALIGN [CENTER | RIGHT] ENDALIGN]
[PAD <n>]
[SPACING <n>]
[SKIP [<n>] ]
[KEY [<key_options>] ]*
[TEXT <text> [<field_options>] ]*
[BUTTON [<field_options>] ]*
[NAME [<name_options>] ]*
```

Global options

The global options describe the default dialog box and dialog element characteristics that apply when it generates.

NOBORDER	Specifies that the dialog box should not include a border.
RESTART <procedure>	Specifies a default procedure to execute when a user selects a button or key.
WIDTH [FULL <n>]	Specifies the width of the dialog - either full screen, or an integer, <n>.
BACKGROUND <color> FOREGROUND <color> <color> [ON <color>]	Specifies the dialog colors, where <color> can be one of: blue, green, red, orange, yellow, white, black, brown, cyan, turquoise, purple, violet, magenta, pink, gray, or grey.
FONTHEIGHT <n>	Specifies a default font size, <n>.
ORIGIN [TOP BOTTOM LEFT RIGHT TOP,MIDDLE CENTER] <row>, <col>	Specifies the position of the dialog. Where MIDDLE, BOTTOM or <row> can be used for vertical positioning, and LEFT, CENTER, RIGHT or <col> can be used for horizontal positioning.

General options

TITLE '<title>'	Defines the dialog title.
ALIGN [CENTER RIGHT] ENDALIGN	Aligns dialog elements.
PAD <n>	Pads <n> spaces before and after text.
SPACING <n>	Inserts <n> spaces between buttons.
SKIP <n>	Skips <n> lines before dialog elements.

Dialog Element (Keys, Text, Buttons, Names) options

```
[KEY [<n> ] <prompt> [ASK | <commands>] [<field_option>] ]
```

Specifies a function key (F1 to F12) dialog element, where:

<n>	Specifies the number of the function key.
<prompt>	Allows a text description of the function key to be included in the dialog.
[TEXT '<text>' [<field_option>]	

Specifies a text element in the dialog, where:

ASK

'<text>' Specifies the text to display.

[BUTTON <prompt> [<field_option>]* [ASK | <commands>] [<field_option>]* ['+' <command>]]

Specifies a button dialog element, where:

<prompt> Specifies the caption to appear on the button.

<command> Specifies command or button information.

[NAME <control_variable> [PROMPT '<text>' [OPTIONAL] [HIDE] [NOPROMPT] [ONE] [MANY] [<validation>]]* [ASK | <commands>] [<field_option>]*]]

Specifies a input field dialog element, where:

<control_variable> Is the name of a control variable in which to store the user response.

PROMPT '<text>' Specifies a label for the input field.

OPTIONAL Indicates that no response is required.

HIDE Indicates that the user response is hidden.

NOPROMPT Used with a validation clause. Bypasses the display of the dialog, and goes straight to the validation option list for user input.

ONE Indicates that the user can only enter a single value.

MANY Indicates that the user can enter a list of values.

<validation> Specifies the validation options. For more information, see the Validation Options section later in this topic.

Field options

Field options specify dialog element characteristics. They can be one or more of the following:

ORIGIN <row>, <column> Places the origin, or upper left corner, of a dialog element at the specified <row> and <column>.

BACKGROUND <color> | FOREGROUND <color> | <color> [ON <color>]

Specifies the dialog element color, where <color> can be: blue, green, red, orange, yellow, white, black, brown, cyan, turquoise, purple, violet, magenta, pink, gray, or grey.

WIDTH <n> Specifies the dialog element width, where n is an integer.

FONTHEIGHT <n> Specifies a font size of the dialog element, <n>.

RESTART <procedure> Specifies a <procedure> to execute on activation of the dialog element.

NORESTART Cancels the global restart procedure for the specified dialog element.

QUIT Exits from the ASK procedure.

Validation options

Validation options describe valid user responses. They also define popup windows with selections the user can access by entering a blank or a question mark (?). Validation options can be one or more of the following:

FROM <list> [HIDE] [SUPPRESS]

A list of possible responses, enclosed in single quotation marks (' ') if they do not comply with rules for Application Server variable names. For example:
FROM Tom, Dick, Harry, '123'.

HIDE indicates that the user response does not appear on the screen.

SUPPRESS prevents the list of available options from appearing on the screen.

HIDE and SUPPRESS are useful for passwords.

<periodicity> PERIOD <daterange>

A standard periodicity, for example, MONTH PERIOD 97-99.

VARIABLE Specifies an existing variable.

MEMBER | SELECTED | INPUT | LEVEL <n>] <dimension>

Specifies an existing dimension. For example:

ASK NAME TEST MEMBER PRODUCT lists members of the PRODUCT dimension

ASK NAME TEST LEVEL <n> PRODUCT lists members of the PRODUCT dimension at level <n>

ASK NAME TEST INPUT PRODUCT
lists the input members of the PRODUCT dimension

ASK NAME TEST OUTPUT lists the output members of the PRODUCT dimension.

TIME | LOGIC | REPORT | PROCEDURE | RECORD | DOCUMENT | SYNONYM | PORTFOLIO |
MULTIMEDIA | LIGHTSHIP | AUDIO | VIDEO | BINARY | DIMENSION

Specifies the name of a set, for example, a logic or dimension set.

Example 1

...This procedure contains an ASK command that prompts the user for a variable

...name and then executes a SHOW VARIABLES command using that response.

...Application Server performs validation to ensure that a valid variable name is entered,

...and will provide a list box of choices if a blank field is entered:

CLEAR Varname

BEGIN

 ASK

 TITLE 'Example 1'

 NAME Varname

 PROMPT 'Enter variable name:'

 VARIABLE

 'SHOW VARIABLES &Varname'

END

Example 2

...The following procedure, REPDEMO, defines three buttons that the user can click to obtain one of

... three reports. The user can use an additional button to exit after Application Server

...executes the desired reports:

BEGIN

 ASK

 TITLE 'Example 2'

 GLOBAL

 RESTART RepDemo

 BUTTON

 'Sales Report' 'SalesRep'

 + 'Expense Report' 'ExpRep'

 + 'Variance Report' 'VarRep'

 + EXIT NORESTART

END

...The plus sign (+) causes buttons to appear across the page. You do not need to repeat the BUTTON

ASK

... keyword because all of the items are buttons. You do not need to use quotation

...marks for the button named Exit because it does not have any embedded spaces or punctuation.

...RESTART causes the menu to reappear after Application Server

...executes each report. It does not appear if the user clicks the Exit button.

Example 3

...This procedure defines a pop-up dialog box that is called from Mainmenu. The dialog box displays

... three fields that prompt you to select variables, a dimension, and a time

...period. If you enter a question mark or a space in a field, a list of items to choose from appears. After

... you select a dimension, another dialog box appears, prompting you to select its members. When all

... the fields are defined, the system returns to Main menu:

Clear Varname, Dimname, Memnames, Pername

BEGIN

 ASK

 GLOBAL

 red ON white

 NOBORDER

 RESTART Mainmenu

 NAME Varname

 PROMPT 'Enter variables: '

 VARIABLE

 'SELECT VARIABLE \&Varname'

 NAME Dimname

 PROMPT 'Enter dimension: '

 DIMENSION

 ASK

 TITLE 'Member Selection'

 NAME Memnames

 PROMPT 'Enter members: '

 MEMBER &Dimname

 'SELECT \&Dimname \&Memnames'

 ENDASK

 NAME Pername

 PROMPT 'Enter period: '

 MONTH PERIOD 96-LATEST

 'SELECT PERIOD \&Pername'

END

Using ASK with control variables

The control variable Varname is used in Example #3. If you select Sales, Expenses in the ASK window, the statement:

SHOW VARIABLES &Varname

would be interpreted as:

SHOW VARIABLES Sales, Expenses

A backslash (\) delays the resolution of a control variable.

In the SELECT command in Example #3, the control variables are evaluated when the commands are executed, not when the ASK command is read.

In addition to creating text for control variables in ASK windows, it is useful to create, display, and clear control variables outside of the ASK window. The commands that allow you to perform these operations are:

SET CONTROL name [']text [']

SHOW CONTROL [pattern]

CLEAR <name> [, ..., <name>]

6.22 ATTACH

Use

ATTACH attaches a database in a specified order.

Syntax

ATTACH <database> [FIRST | LAST] [EXCLUSIVE | SHARED | READ]

Parameter	Description
<database>	Name of the database to attach.
FIRST	Attaches the database as the first database. This is the same as entering the command USE <database> RETAIN.
LAST	Attaches the database as the last database (the default).
EXCLUSIVE	Indicates that only you can read and update the database (the default).
SHARED	Indicates that all users can read and update the database. Users can update sets within the database, but they cannot change the structure of the database or data within it.
READ	Indicates that all users can read the database, but cannot update it.

Example:

...This example attaches the Juice database:

ATTACH Juice

6.23 ATTRIBUTE

Use

ATTRIBUTE opens the Attribute editor. You use an attribute set to view, create, or edit attributes for a dimension. Entering a CONSTRUCT ATTRIBUTE command constructs an attribute and automatically creates an attribute set. You can also manually create the attribute set by following the syntax described here.

Syntax

ATTRIBUTE <setname>

BY <dimension> [RESULT] [LEVEL] [LABEL "<label>"] [CUSTOM <number>]

VARIABLE <attribute>

<member> ['<label>']

,<member> ['<label>']

ATTRIBUTE

```

, ...
[VARIABLE <attribute>
  <member> ['<label>']
  ,<member> ['<label>'] ]
, ...

```

Parameter	Description
<setname>	Name of the attribute set to create or edit.
<dimension>	Name of a dimension that this attribute(s) is associated with.
<attribute>	Name of the attribute for this dimension. An attribute set can have any number of attributes.
<member>	Member of the attribute. Specify multiple members separated by commas.
<label>	Optional label for the attribute member. If your label contains single (') or double (") quotation marks, you need to encapsulate the member label in a different quoting character, for example, "Don't try this at home".
	These keywords can be added to an Attribute set following the BY <dimension> line:
RESULT LEVEL	<p>Ensures that across/down views are not inadvertently restricted based upon an attribute, and it also assigns a default level name (which is the name of the attribute) to the level inputs of that attribute.</p> <p>If you do not explicitly select a dimension member, Application Server makes a determination about the dimension member to use. If the dimension has a result member, that is chosen. If the dimension does not have a result member (as is usually the case with attribute dimensions), then the first member is chosen.</p> <p>For example, assume that Total Product equals \$100,000. If you include a Color attribute set but do not specify RESULT LEVEL, you may only see \$2,000, because only \$2,000 is associated with products with the attribute of Green (the first member of the Color attribute set).</p>
RESULT LABEL "<label>"	<p>Use RESULT LABEL "<label>" to specify a label for the Result level member.</p> <p>For example:</p> <pre> BY CUSTOMER RESULT LABEL "Result Member" CUSTOM 5 </pre> <p>shows "Result Member" as the label for the Result level member, and:</p> <pre> BY CUSTOMER RESULT LABEL "%s Total" CUSTOM 5 VARIABLE PARENT </pre> <p>shows "PARENT Total" as the label for the Result level member.</p> <p>Note: If you do not specify RESULT LABEL "<label>", by default, "TOTAL <member>" is displayed as the label for the Result level member.</p>
CUSTOM <number>	Enables an attribute for User-Defined Hierarchies, where <number> is the number of User-Defined Hierarchies that can be defined for each attribute in the set.

Set editor example

...This example shows syntax for creating an attribute using the set editor. The Color attribute is ... associated with the Product dimension and has members called Pink, Red, Green, and Mauve:

ATTRIBUTE Prod_att

BY Product

VARIABLE Color

Pink

,Red

,Green

,Mauve

Checking that the attribute was constructed properly

When you construct an attribute, it creates an attribute set containing the constructed attribute dimensions. It also creates attribute variables for each dimension based on the fields you referenced from the source. You should check the attribute set, the attribute dimensions, and the attribute variables to make sure they were constructed properly.

To check the attribute dimensions

Enter:

EXHIBIT ADIMENSION

To check the attribute variables

Enter:

SHOW VARIABLES

To check the attribute set

1. Enter:

DIRECTORY ATTRIBUTE

Application Server displays the attribute sets. Optionally, you can enter SHOW ATTRIBUTE.

2. Go to the attribute set:

ATTRIBUTE <setname>

3. Check the structure. Multiple VARIABLE statements mean that this attribute set has multiple attribute dimensions and was constructed from one source file. If the attribute information was in separate source files, you will have separate attribute sets.

BY <dimension>

VARIABLE <field>

 <member> '<label>'

 , ...

 ,<member> '<label>'

VARIABLE <field>

 ,<member>, ...

CLASS <name> <member>, <member>, ...

where <dimension> is the name of the entity dimension this attribute is associated with.

where <field> is the name of the attribute. Application Server derives them from the source column name.

where <members> are the attribute members. Application Server derives them from the values in the source column.

where <labels> are the attribute member labels. They are optional. Application Server derives these labels from the values in the source column. If your label contains single (') or double (") quotation marks, you need to encapsulate the member label in a different quoting character. For example: "Don't try this at home".

6.24 BEGIN-END

Use

With BEGIN-END you can spread long commands over several lines in a procedure. When Application Server finds BEGIN, it treats successive lines as part of one command until it finds END, at which point Application Server executes the command.

Syntax

```
BEGIN
    .
    . <long Application Server command>
    .
END
```

6.25 BYPERIOD-ENDBYPERIOD

Use

BYPERIOD-ENDBYPERIOD forces all enclosed statements to execute for period 1 before moving to period 2. Application Server executes all statements for period 2 before moving to period 3, and so on.

Note: You cannot nest BYPERIOD-ENDBYPERIOD statements.

Syntax

```
BYPERIOD
    .
    . <logic_statements>
    .
ENDBYPERIOD
```

Example

...This user-defined function calculates Net Terminal Value (NTV). Application Server computes a
...new value of NTV for each period, based on the previous value of NTV, and then inputs Forecast
...and Rate. Without BYPERIOD-ENDBYPERIOD, Application Server would add NTV to forecast [1] for
... all periods, multiply that result by rate [1] for all periods, and then store the results in NTV.

INPUT Forecast, Rate

RESULT SCALAR NTV

NTV = 0

BYPERIOD

NTV = Rate[-1] * (NTV + Forecast[-1])

ENDBYPERIOD

NTV = NTV + VALUE(Forecast, NCASES())

6.26 CALCULATE

Use

CALCULATE evaluates an expression and assigns the result to a variable, or evaluates a group of expressions in a logic set and makes multiple assignments. In either case, CALCULATE creates new variables from existing ones.

Syntax

CALCULATE

```
{ <variable> = <expression> [FULL] [TRACE [<n>]] [DATA] [GROUP | NOGROUP] [ RESOLVE ]
  [SPARSITY] |
  <logic> [<periodicity>] [PERIOD <daterange>] [FULL] [TRACE [<n>]] [DATA] [ RESOLVE ] |
  <variable> = <expression> [FULL] [TRACE [<n>]] [DATA] [ RESOLVE ] [ FASTMODE | SPARSITY <n> ]
}
```

Parameter	Description
<variable>	Name of the result variable. Variable names that use special characters should be in single quotation marks (' ').
<expression>	Arithmetic expression or an integer.
<logic>	Name of a logic set containing one or more expressions. If you do not specify a name, Application Server uses the default logic set (if defined). Otherwise, it uses the last logic set edited.
<periodicity>	Performs the calculation at one of the following periodicities: yearly, semiannually, quarterly, monthly, lunar, bimonthly, weekly, biweekly, daily, or hourly.
PERIOD	Limits the calculation to a specified date range.
<daterange>	A range of dates separated by a dash (-).
FULL	Calculates the logic set at all levels in a dimension.
TRACE	Displays the names of the dimension member combinations for which variables are being calculated.
<n>	Displays the name of every <n>th dimension member combination calculated. The default, 1, displays the name of every dimension member combination calculated.
DATA	<p>If you are calculating data into an output level, DATA marks the data as read-in data instead of calculated data. During consolidation, Application Server treats calculated data differently from read-in data. If you mark the calculated data as DATA, Application Server does not overlay values in the output level in quadrants specified with CHANGE UPDATE NOOVERWRITE.</p> <p>Note: The DATA keyword applies only when new output series are created. For any existing output series which are modified by the CALCULATE, the DATA keyword is not applied.</p> <p>For example, say you are projecting this year's budget data into next year, modified by some percentage. This would require a calculation in which budget data is entered at a consolidated level. If you will not be allocating this calculated data down to a lower level, you must tell Application Server to treat it as read-in data. You would enter a CALCULATE DATA command to have Application Server treat the calculated data as read-in data.</p> <p>If you allocate the budget data down to the detail level, do not use DATA since Application Server should treat the data as calculated data. During consolidation, Application Server would add the data correctly.</p>
<VARIABLE>	Calculates all selected variables.
GROUP	Attempts to partition the output series better during a calculation. Time-series are clustered by quadrant in the Application Server database. All series for quadrant N go into the same database block. There is no mixing of series from different quadrants in the same database block. The larger the calculation, the more effective this is. If you do not specify the GROUP or NOGROUP keyword, then GROUP is used by default.
NOGROUP	Does not cluster time-series by quadrant in the Application Server database during a calculation.
RESOLVE	If you specify this parameter, the procedure or logic automatically recompiles before execution. This ensures that the current values of control variables in the set are used.

CALCULATE

SPARSITY <n>	Populates every <n>th cell of input data. Note: Do not use SPARSITY with FASTMODE.
FASTMODE	When you calculate an expression based on variables with different dimensionalities, fast calculation is now the default. However, if your expression contains a subscripted expression, for example SALES IN PRODUCT TOTAL PRODUCT, Application Server by default will not use fast consolidation. Use the FASTMODE keyword if you want to force fast consolidation in this instance. Note: Do not use SPARSITY with FASTMODE.

Examples

...This example creates the variable Sales_Revenue, which has the dimensions, periodicity, ...and date range indicated by the variables Price and Number_of_Cases. Number_of_Cases ...is dimensioned by Product and Region, and Price is dimensioned by Product. Therefore, ...Sales_Revenue is dimensioned by Product and Region:

CALCULATE Sales_Revenue = Price * Number_of_Cases

...This example shows a conditional calculation. The revenue expression varies based on ...the volume, taking into account discounts:

BEGIN

CALCULATE Sales_Revenue

IF Number_of_Cases LT 1000

= Price * Number_of_Cases

OTHERWISE

= 0.9 * Price * Number_of_Cases

END

...Set a time period and calculates daily results according to the logic set FLOW:

CALCULATE flow PERIOD 00/1 daily

...See how Application Server assigns values to several variables in one statement:

CALCULATE x=1 y=2 z=3

...This example sets all selected variables equal to 4:

CALCULATE <VARIABLE> = 4

...This example calculates sales data as read-in data at the output level:

ROLLUP Sales

ADD EVERYBODY

CHANGE 5 UPDATE CONSOLIDATE Product, Region

END

SELECT Sales

SELECT Product OUTPUT

SELECT Region OUTPUT

SELECT Manufacturer INPUT

SET PERIOD 1/00

CALCULATE Sales = 4 FULL DATA

6.27 CHANGE DATABASE (Supervisor)

Use

A Supervisor command that changes a database definition in MASTERDB.

Syntax

```
CHANGE DATABASE <database>
    [ EXHIBIT | NOEXHIBIT ]
    [ ADD <num> PARTITIONS [SIZES <sizes>] [PREFORMAT] ]
    [ PROTECTION <key> ]
    [ READ <read> ]
    [ UPDATE <update> ]
    [ DISPOSITION {OFFLINE | ENABLED | DISABLED} ]
    [ MAXACCESS { READ | UPDATE } ]
    [ USAGE [EXCLUSIVE | READ | SHARED] [RETAIN] ]
    [ VARIABLES <var> ]
```

Parameter	Description
<database>	The name of the database to change.
EXHIBIT	Specifies that the database details will always be displayed by the EXHIBIT DATABASES command. This is the default value.
NOEXHIBIT	Specifies that the database details will only be displayed by the EXHIBIT DATABASES command if you use the command with the HIDDEN keyword.
ADD <num> PARTITIONS	The number of partitions to add to the database. See Creating a database with multiple partitions for more information.
SIZES <sizes>	One or more numbers that identify the size of each new partition in blocks. You must separate each size with a comma. You can specify <sizes> as the number of blocks (for example, 20,000), as a multiple of 1,000 with a K suffix (for example, 20K), or as a multiple of 1,000,000 with an M suffix (for example, 1M). By default, Application Server divides the partition sizes evenly based on the number of blocks. If you specify fewer sizes than the number of partitions, Application Server divides the remaining blocks equally among the remaining partitions. If you increase the partitions to a size greater than the database, you should extend the database. You cannot change the size of an existing partition.
PREFORMAT	Application Server writes to all the blocks in the partition to ensure that the space is preallocated.
PROTECTION <key>	A protection key, where <key> is an alphanumeric name up to 16 characters.
READ <read>	One or more read access keys separated a comma. Each read access key is an alphanumeric name up to 16 characters. You can set both a READ key and an UPDATE key on a database. Use the Supervisor REMOVE command to remove a read or update access key.
UPDATE <update>	One or more update access keys separated a comma. Each update access key is an alphanumeric name up to 16 characters.
DISPOSITION	Specifies the disposition of the database. You can specify OFFLINE, ENABLED, or DISABLED.
MAXACCESS	Specifies the maximum access level at which users can attach to the database. You can specify READ for read access or UPDATE for update access.
USAGE [EXCLUSIVE READ SHARED] [RETAIN]	Indicates the access mode when a user enters a USE or ATTACH command without specifying a usage parameter. You can specify READ for read access, SHARED for shared access, or EXCLUSIVE for exclusive access (the default). In addition, you can specify the RETAIN keyword to keep the current database attached.
VARIABLES <var>	The maximum number of variables the database can contain. The default value is 1,000, and the maximum is 10,000.

CHANGE USER (Supervisor)

Example 1

...This example changes the definition of the database BUDGET:

CHANGE DATABASE budget MAXACCESS READ

Example 2

...The ACCNT database already has three partitions. The following command adds two partitions. One ... is 40,000 blocks long, and the other is 20,000 blocks long. Because the partitions are larger than ... the number of blocks specified for the database, Application Server extends the database.

DETACH Accnt

CHANGE DATABASE Accnt ADD 2 PARTITIONS SIZES 40K, 20K

...The output of this command would be:

Warning: The Number of Blocks Specified in the PARTITION SIZES (120000) for Database ACCNT is Larger than the BLOCKS specified.

Database: ACCNT

Maxaccess: UPDATE Disposition: ENABLED

Current State: AVAILABLE Protection Key: None

Last user: Last access:

Blksize: 8192 Observations 500

Maxblocks: 60000 Number Free: 59990

Database ACCNT Partition Information.

Number of Partitions: 5

Blocksize: 8192

Date Last Opened for Update: Thu Feb 9 15:34:10 1999

Date Last Closed After Update: Thu Feb 9 15:34:10 1999

Partition	Size	Name
1	20000	ACCNT01
2	20000	ACCNT02
3	20000	ACCNT03
4	40000	ACCNT04
5	20000	ACCNT05

USE Accnt

EXTEND Accnt 60000

6.28 CHANGE USER (Supervisor)

Use

CHANGE USER is a Supervisor command that changes a user record in MASTERDB.

Syntax

```
CHANGE USER <user>
      [ USEDDB <usedb> ]
      [ USAGE [EXCLUSIVE | SHARED | READ] ]
```



```

[ WORKDB <workdb> ]
[ BLOCKS <blocks> ]
[ BLKSIZE <blksize> ]
[ BUFFERS <buffers> ]
[ SECURITY [0 | 99] ]
[ LOGIN [ON | ENABLED | DISABLED | PAUSED] ]
[ EXPIRATION <date> ]
[ PASSWORD <password> ]
[ MAXLOGIN <lognum> ]
[ ACCESS <key> ]

```

Parameter	Description
<user>	The name of the user record to change.
USEDDB <usedb>	Specifies the database the user is attached to after logging in. If you omit this parameter, Application Server attempts to attach the user to a database with the same name as the user name.
USAGE [EXCLUSIVE SHARED READ]	Specifies the mode in which the default database is opened. You can specify READ for read access, SHARED for shared access, or EXCLUSIVE for exclusive access. This setting is used when logging in.
WORKDB <workdb>	Specifies the file name Application Server should use to create the Work database. You can specify an alphanumeric name of up to eight characters. Do not include a path; Application Server creates all Work databases in the home directory. The default is DB<nnnn>. The user has read and update access to this database.
BLOCKS <blocks>	The number of blocks that should be allocated to the Work database. The minimum is 2000 blocks and the default is 2000 blocks.
BLKSIZE <blksize>	The size of blocks in the Work database in bytes. You can specify 512, 1K, 2K, 4K, 8K, or 16K. The default is 8K.
BUFFERS <buffers>	The number of buffers to use. The minimum is 20 and the default is 2000.
LIBRARY <libdb>	The library database for the user. You can specify an alphanumeric name up to 16 characters. The default is TBDB.
SECURITY [0 99]	The user's security level. Specify 0 for a normal user or 99 for a user with Supervisor privileges. The default is 0.
LOGIN [ON ENABLED DISABLED PAUSED]	Specifies the state of the user's login. Specify ON, ENABLED, DISABLED, or PAUSED.
EXPIRATION <date>	The date on which the user's account expires. After this date, the user will be unable to access Application Server.
PASSWORD <password>	The user's password. Specify an alphanumeric name with up to 12 characters. The first character must be alphabetic, not numeric.
MAXLOGIN <lognum>	The number of times the user can log in to Application Server concurrently. Specify a number between 1 and 32,000. The default is 255. You might want to allow a user multiple logins when many users within a User-Defined Hierarchy or department will use the same login ID to access Application Server.
ACCESS <key>	Specifies an access key that is required to access a database in either read or update mode.

Example

...This example changes the record for user Fin2:

CHANGE USER Fin2 SECURITY 99

6.29 CHECKPOINT

Use

CHECKPOINT defines how and when to update a database.

Syntax

```
CHECKPOINT
  { [BACKUP]
    [FREEZE [CONTINUE] [NOBACKUP]]
    [SHOW]
    [SYNCHRONIZE [0|1]]
    [UPDATE [<n>]]
    [WORKDB <n>] }
```

Parameter	Description
BACKUP	Restores the database to the state it was in when you entered the last CHECKPOINT FREEZE command, or to the state it was in when Application Server last automatically updated it according to the value of <n> in CHECKPOINT UPDATE <n>.
FREEZE	Updates the database to its present state and then suppresses automatic updating of the database after every command, even when the Use database is in READ mode. Application Server does not update the database, data, and sets again until you issue CHECKPOINT UPDATE.
CONTINUE	Keeps you in CHECKPOINT FREEZE mode automatically after an error occurs. Application Server rolls back your database to the point it was at when you entered the original CHECKPOINT FREEZE CONTINUE command, but you do not have to re-enter the command to stay in CHECKPOINT FREEZE mode. You remain in that mode until you enter a CHECKPOINT BACKUP or a CHECKPOINT UPDATE command.
NOBACKUP	Keeps you in CHECKPOINT FREEZE mode automatically after an error occurs, but, unlike the CONTINUE keyword, does not roll back your database. You remain in CHECKPOINT FREEZE mode until you enter a CHECKPOINT BACKUP or a CHECKPOINT UPDATE command.
SHOW	Outputs a line of four values. The first is the current CHECKPOINT FREEZE status, where 0 means that it is not in a CHECKPOINT FREEZE and 1 means that it is in a CHECKPOINT FREEZE. The second value represents the value specified in a CHECKPOINT UPDATE. The third value represents the value specified in a CHECKPOINT SYNCHRONIZE. The last value represents the value specified in a CHECKPOINT WORKDB.
SYNCHRONIZE [0 1]	Enables "soft" writes to the work database. By default, Application Server performs soft writes to the work database. If you are already using checkpoint effectively in applications, this feature may not result in a performance gain. If you are not yet using checkpoint, CHECKPOINT SYNCHRONIZE could reduce end-user response time by up to 30%.
UPDATE	Enables automatic updating of the database and, if CHECKPOINT FREEZE is in effect, brings the database to its current state.
<n>	If $n > 1$, Application Server updates the database after the successful completion of every <n> commands. Otherwise, it updates the database after the successful completion of every command.
WORKDB <n>	Controls the number of commands that Application Server waits before it flushes its buffers to the work database. <N> represents the number of commands. The default value is 1.

Example: CHECKPOINT

...In this example, the database has three logic sets: PROFIT, REVENUE, and SALES. CHECKPOINT
... FREEZE is entered and the database is not automatically updated. REMOVE removes the logic set

... PROFIT from the database. CHECKPOINT BACKUP then restores the database to its state during the ... initial FREEZE. PROFIT appears in the directory list.

CHECKPOINT FREEZE

DIRECTORY LOGIC

LOGICS

PROFIT

REVENUE

SALES

REMOVE LOGIC profit

DIRECTORY LOGIC

LOGICS

REVENUE

SALES

CHECKPOINT BACKUP

DIRECTORY LOGIC

LOGICS

PROFIT

REVENUE

SALES

6.30 CLASS (Dimension or CONSTRUCT)

Use

CLASS is a Dimension command or CONSTRUCT statement that defines a <class>, or group of dimension members, within a dimension. Classes are useful to define small groups or subsets of data. You can select members by their class name instead of selecting them individually.

Syntax

CLASS <class> <member> [... <member>]

Parameter	Description
<class>	Name of the class up to 24 alphanumeric characters.
<member>	Name of a previously defined member. You can use an asterisk (*) in a member name as a wildcard that indicates that any character can occupy that position or any of the remaining positions.

Example

...Consider the following statements defined in a Dimension editor:

INPUT Coca_Cola, Diet_Cola, Cherry_Coke, Orange_Fanta, Sprite

CLASS Cola Coca_Cola, Diet_Cola, Cherry_Coke

...This statement selects a class:

SELECT Product Cola

...This statement uses a class in a logic statement:

WHEN Product IS Cola

...These statements define a new dimension member to be the sum of an existing class:

CHG()

Total_Cola = Cola

Total_Inputs = Input

6.31 CHG()

Use

CHG returns the absolute differences between the current period and the same period a specified interval (for example, a year or month ago).

Syntax

CALCULATE <result> = CHG(<variable> [, <period>])

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable to evaluate. Variable names that use special characters should be in single quotation marks (' ').
<period>	Period in which to evaluate: year (default), quarter, month, or week.

6.32 CLEAN WORK DATABASES (Supervisor)

Use

CLEAN WORK DATABASES is a Supervisor command that safely removes orphaned Work databases by only removing databases that it can open exclusively (that are not in use by any other user) and which it can verify are true Work databases.

You can use CLEAN WORK DATABASES on any Work database created in 7.5 SP10 or higher. If you want to remove a Work database created in an earlier release than 7.5 SP10, you must manually remove the Work database.

Syntax

CLEAN WORK DATABASES [TRACE] [PREVIEW] [<pattern>]

Parameter	Description
TRACE	Specifies that you want to output trace information listing each database that is removed.
PREVIEW	Specifies that you want to preview a list of databases that will be removed without actually removing them.
<pattern>	A standard pattern with or without wildcards to remove only databases that match this pattern, for example, DB*.

Examples:

Supervisor> clean work preview DB*

DB40510

DB40575

Supervisor> clean work trace *75*

DB40575

Supervisor> clean work preview DB*

DB40510

6.33 CLEAR

Use

The CLEAR command clears User-Defined Hierarchies from a dimension. You can also use the CLEAR command to remove control variables.

Notes:

When you clear a dimension's User-Defined Hierarchies, Application Server selects all of the dimension's members.

If this User-Defined Hierarchy has been saved using the SAVE CUSTOM command that includes the PUBLIC or PRIVATE keywords, then the compiled dimension set for this User-Defined Hierarchy remains saved in the CGLIB database and will be restored again during the next RESTORE CUSTOM command.

If you want to completely remove the User-Defined Hierarchy from your database, including the compiled dimension set, then use the REMOVE CUSTOM command instead.

Syntax

```
CLEAR {<dimension> [USING [<owner>].<setname>]
      {<dimension> <user_defined_hierarchy> [,<user_defined_hierarchy>,...]}
      {CUSTOM <dimension> [,<dimension>,...]}
      {<control_variable>}}
```

Parameter	Description
<dimension>	Name of the dimension whose User-Defined Hierarchies you want to clear. If you use CLEAR <dimension> without specifying any other keywords, then all the User-Defined Hierarchies for that dimension will be cleared except those created using SAVE CUSTOM PUBLIC or SAVE CUSTOM PRIVATE.
USING	Clears User-Defined Hierarchies from a dimension that has the PUBLIC or PRIVATE keywords applied in a SAVE CUSTOM command.
<owner>	Name of the owner of this User-Defined Hierarchy. If you omit the owner name, the current Application Server user is used. Only the administrator or owner of a User-Defined Hierarchy can clear a User-Defined Hierarchy.
<setname>	Name of the procedure set to clear, that was created during the SAVE CUSTOM command that included a PRIVATE or PUBLIC keyword. This clears all the User-Defined Hierarchies for the dimension.
<user_defined_hierarchy>	Name of the User-Defined Hierarchy you want to clear from the session. Specify more than one User-Defined Hierarchy, separated by commas, to clear multiple User-Defined Hierarchies.
<control_variable>	Name(s) of control variable(s) you want to remove. To indicate more than one control variable, separate them with commas: CLEAR variable1,variable2

Examples:

...This example removes two control variables, Logset and Dimset:

```
CLEAR Logset, Dimset
```

...This example removes all control variables:

```
CLEAR *
```

...This example clears all User-Defined Hierarchies from the Product dimension:

```
CLEAR product
```

...This example clears the Monitor User-Defined Hierarchies from the Product dimension:

```
CLEAR product monitor
```

6.34 CLEAR STATUS

Use

CLEAR STATUS resets a user's status to the original login state. CLEAR STATUS resets the ACROSS DOWN list to the default, which is ACROSS Time DOWN variables. It also clears any dimension and variable selections and clears the period and periodicity settings.

Syntax

CLEAR STATUS

Example: CLEAR STATUS

...This example shows the current status, clears the status, and then shows the updated status:

STATUS

...The output of this command would be:

Across List: # Selected

 PRODUCT 3 of 19 Processor Chips, Color Monitors, Hard Drives

 REGION 22 of 36 New York, Chicago, Washington, Denver...

 TIME

Down List:

 TYPE 3 Actual, Budget, Variance

 VARIABLES 2 Cost of Sales, Sales

Periodicity MO Period 1995/1/2 - 1995/12/31 Earliest 04 Jan 95 (MO)

Latest 30 Oct 95 (MO)

Attached Databases: DEMO User: ADMIN

445 Fiscal Year Starts in January, Weeks Begin Monday

...This example shows the output when the status is cleared using the

...CLEAR STATUS command, and the STATUS command is entered.

CLEAR STATUS

STATUS

...The output of this command would be:

Earliest 04 Jan 95 (MO)

Latest 30 Oct 96 (MO)

Attached Databases: DEMO User: ADMIN

445 Fiscal Year Starts in January, Weeks Begin Monday

6.35 COLHEADING-ENDCOLHEADING (Report)

Use

COLHEADING-ENDCOLHEADING is a Report command that defines a block of column headings.

Syntax

COLHEADING

. <column heading specifications>

ENDCOLHEADING

Example

...In this block, the across dimension is Country:

COLHEADING

TEXT OVER 2 "United", OVER 3 "United"

TEXT OVER 2 "Kingdom", OVER 3 "States"

TEXT OVER 2 "=====", OVER 3 "=====

ENDCOLHEADING

...Application Server will produce the following column headings:

United	United
Kingdom	States
=====	=====

6.36 COMPILE

Use

COMPILE compiles a set or an attribute set.

Generally, when you exit from a set editor, Application Server automatically compiles the set. You must compile a set at the following times:

When you construct a dimension or attribute from the ACCESS subsystem. (If you construct a dimension or attribute in the Set editor, Application Server automatically compiles it when you exit from the editor.)

When you copy a file from an external source into an Application Server set.

When you recover a set using the RECOVER command.

6.36.1 Compiling dimensions whose members have multiple parents

When you compile a dimension (either standard dimension or one used for a User-Defined Hierarchy), the system automatically searches for multiple counting issues that arise if the dimension's members have multiple parents. During aggregation, the system corrects multiple counting issues by creating adjustments to eliminate the multiple counts. The adjustments are used during a CONSOLIDATE. No data values are changed until a CONSOLIDATE command is issued to reconsolidate data.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC1 Selected

The dimension compiler detected multiple counting issues in Dimension PRODUCT.

Adjustments have been made to internal aggregation rules to correct these issues.

COMPILE ATTRIBUTE

Automatic multiple counting corrections can only be performed on measures where a smart fast consolidate can be used (i.e., all of the dimensions have either simple additive or subtractive consolidations) – i.e. the measure can be used with the ROLLUP editor.

Automatic multiple counting corrections will **not** be performed on a measure if:

- Any of its dimensions have non additive calculations
- The measure is an INTEGRAL measure
- The user uses the NOCORRECTIONS keyword on COMPILE DIMENSION. This turns off the automatic multiple counting corrections for that dimension.
- All aggregations of a member that cause multiple counting must be of the same “sign” i.e. all additive or all subtractive.

For example:

System> COMPILE DIMENSION product

1 Member Rolls Into Multiple Outputs

All 6 Members of PRODUCT;MC3 Selected

Dimension PRODUCT contains both positive and negative sums.

Compiler could not check for multiple counting

Note: If you have created an attribute with hierarchies and you are issuing the COMPILE DIMENSION command to compile the hierarchical attribute, the system checks for multiple counting issues in the hierarchical attribute dimension just like it does for standard dimensions. At run time, any dynamic aggregations performed will automatically adjust the aggregations to correct any multiple counting issues in an regular dimensions or hierarchical attribute dimensions in the view.

For information, see [Multiple Counting in Dimensions](#).

Syntax

COMPILE <settype> <setname> [NOCORRECTIONS]

Parameter	Description
<settype>	Type of set: dimension, attribute, logic, procedure, report, synonym, or time.
<setname>	Name of the set to compile.
NOCORRECTIONS	When compiling a dimension, this setting specifies that the system should perform multiple counting. Application Server will not search for multiple counting issues and all subsequent aggregations performed on those dimensions are aggregated exactly as defined in the hierarchy with no corrections.

6.37 COMPILE ATTRIBUTE

Use

Use the COMPILE ATTRIBUTE command to compile an attribute set. When you use this command, Application Server automatically creates variables and saves any data they might currently contain.

Syntax

COMPILE ATTRIBUTE <name>

Parameter	Description
<name>	The name of the attribute set.

Example

...Copy an external file extrep into a report set in the DEMO database and then compile the report set:

COPY Report Extrep;EXTERNAL Newrep;DEMO

COMPILE newrep

...Construct a dimension in the ACCESS subsystem, exit from ACCESS, and compile the dimension:

ACCESS EXTERNAL

USE sourcefile

BEGIN

DESCRIPTION statements

END

BEGIN

CONSTRUCT statement

END

COMPILE DIMENSION Product

...This example compiles the Product attribute:

COMPILE ATTRIBUTE Product

Example

...This example compiles the Product attribute:

COMPILE ATTRIBUTE Product

Adding a class - Example

...This example constructs an attribute set called Prodatt because it is associated with the Product

... dimension. The attribute is Color, and has two classes of colors, Pastels and Primary colors.

ACCESS LSLINK

CONNECT dbase

SELECT * from Product

CONSTRUCT ATTRIBUTE Prodatt BY Product VARIABLE color CLASS Shades

END

COMPILE ATTRIBUTE Prodatt

6.38 CONSOLIDATE

Use

CONSOLIDATE consolidates variables.

When you compile a dimension, the system automatically searches for multiple counting issues that arise if the dimension members have multiple parents. During aggregation, the system corrects multiple counting issues by creating adjustments to eliminate them. The adjustments are used during a CONSOLIDATE or any dynamic runtime consolidations. No data values are changed until a CONSOLIDATE command is issued to reconsolidate data. For information, see [Multiple Counting in Dimensions](#).

CONSOLIDATE

Syntax

CONSOLIDATE [<logic> | <report> | <variables> [GROUP | NOGROUP]] [PERIOD <daterange>]

Parameter	Description
<logic>	Name of a logic set whose variables you want to consolidate.
<report>	Name of a report whose variables you want to consolidate.
<variables>	One or more variable names separated by commas. If you do not specify a variable name, Application Server consolidates the currently selected variables. Variable names that use special characters should be in single quotation marks (' ').
GROUP	Attempts to partition the output series better during a consolidation. Time-series are clustered by quadrant in the Application Server database. All series for quadrant N go into the same database block. There is no mixing of series from different quadrants in the same database block. The larger the consolidation, the more effective this is. If you do not specify the GROUP or NOGROUP keyword, then GROUP is used by default.
NOGROUP	Does not cluster time-series by quadrant in the Application Server database during a consolidation.
PERIOD	Consolidates data within the specified date range.
<daterange>	A range of dates separated by a dash (-).

Examples

...This example consolidates the variables Sales and Expenses:

CONSOLIDATE Sales, Expenses

...This example consolidates all variables used in the report BUDREP:

CONSOLIDATE budrep

Creating the variables, reading data into them, and consolidating

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

This procedure creates the sales, broker_com, and cases variables and reads data into the variables:

...Create the variables

CREATE sales, broker_com, cases BY product, customer

...Select the items and set up the view

ACROSS var DOWN product, customer, time

SELECT var BY product, customer

SELECT product input

SELECT customer input

SET PERIOD 1998

...Read data into the variable

ACCESS LSLINK

CONNECT dbase

SELECT * FROM transact

STATUS

PEEK ONLY 10

LSS CREATE product = code

```

LSS CREATE customer = account_co
LSS CREATE time = SUBSTR(date,1,2) + "/" + SUBSTR(date,3,2)
SELECT * FROM transact
READ ADD
END
...Consolidate
SELECT Product
SELECT Customer
SELECT var BY product, customer
ROLLUP sales
ADD EVERYBODY
END
CONSOLIDATE

```

The result of a SHOW VARIABLES command looks like this:

PRODUCT:

Variable	Type	Items	Date Range
FLAVOR	AT [8]	CO	
PACK_CNT	AT [3]	CO	

CUSTOMER, PRODUCT:

Variable	Type	Items	Date Range
BROKER_COM	NU	12 MO	Jan 98 - Dec 98
CASES	NU	12 MO	Jan 98 - Dec 98
SALES	NU	12 MO	Jan 98 - Dec 98

The best way to load two or more years of data, is to load all the data at one time. In this way Application Server puts all the data into contiguous space. This speeds up the load times and keeps the space considerations down.

6.39 CONSTRUCT (Access)

Use

CONSTRUCT is an Access command that creates or updates a dimension based on data in an external source. You can use CONSTRUCT to construct a dimension with the User-Defined Hierarchy capability.

Notes:

Use the CONSTRUCT ATTRIBUTE command to construct attributes without hierarchies. If you have, or want to create hierarchical attributes associated with a dimension, you must construct a separate dimension for each attribute using a separate CONSTRUCT command.

When constructing a dimension, if a member is referenced in the input file more than once, the label that is eventually used for that member is the last label encountered in the input file.

You can use the OVERWRITE keyword to rebuild an existing dimension based on the information in the external data source. Consolidation rules will be rewritten and members that are no longer part of the external source will be deleted from the dimension. Use of the OVERWRITE keyword generally requires a reconsolidation of the model.

CONSTRUCT (Access)

Full syntax for a dimension with one hierarchy

```
CONSTRUCT <dimension> LEVEL <field> [... <field>]
  [ LABEL <field> [... <field>] ]
  [NORESULT]
  [CLASS <field> [MEMBER <field>]
  [... CLASS <field> [MEMBER <field>] ]
  [NOTOTALS] ]
  [PREFACE {" [CUSTOM <number>]
  [KEY BOTH]
  [ALLOCATE [UPDATE] <input>, <output>, <result> ] "}
  [ARRAYSIZE <n>]
  [REPLACE CALCULATIONS]
  [OVERWRITE]
  [REPLACE CLASSES]
  [SINGLE]
  [NOINPUT]
```

Full syntax for a dimension with multiple hierarchies

```
CONSTRUCT <dimension>
  .
  HIERARCHY hierarchy LEVEL <field> [... <field>]
  {APEX apex | NOAPEX}
  [LABEL <field> [...<field> ]
  .
  [CLASS <field> [MEMBER <field>]
  [... CLASS <field> [MEMBER <field>] ]
  [NOTOTALS] ]
  [PREFACE {" [KEY BOTH]
  [ALLOCATE [UPDATE] <input>, <output>, <result> ] "}
  [ARRAYSIZE <n>]
  [REPLACE CALCULATIONS]
  [OVERWRITE]
  [REPLACE CLASSES]
  [SINGLE]
```

Note: You must compile a dimension after you construct it.

Short syntax for hierarchical attributes

```
CONSTRUCT <hierarchy_name> LEVEL <field>,..., <field>
  [ LABEL <field> [... <field> ] ]
  PREFACE "BY <dimension> "
```

Short syntax for dimensions with User-Defined Hierarchies

```
CONSTRUCT <hierarchy_name> LEVEL <field>,..., <field>
  [ LABEL <field> [... <field> ] ]
  PREFACE "CUSTOM <number> "
```

Parameter	Description
<hierarchy_name>	Name of the dimension to create as a container for an attribute's hierarchies.
<dimension>	Name of the dimension to create or update. A dimension name cannot be longer than 96 bytes.
PREFACE	Prefaces the resultant dimension with the specified option. You must enclose the keywords in quotation marks. This option, followed by a quoted string, allows you to put anything in either a dimension or an attribute set.
CUSTOM <number>	Specifies that the dimension will have User-Defined Hierarchies, where <number> is the number of custom User-Defined Hierarchies that the dimension can have.

LEVEL <field>	<p>Specifies the data source fields to use when building the levels of the dimension or the levels in a dimension's hierarchy. You must separate all fields with commas.</p> <p>Each field represents a level. Field values are the members of the dimension. You can specify up to 20 levels in a dimension.</p> <p>You define the lowest level first. The first field you specify contains the lowest level of data, the input members. The next field you specify contains the next higher level of data, the output members. The next field contains the next higher level, and so on.</p> <p>Notes:</p> <ul style="list-style-type: none"> • The field's values become member names when constructed; member names can be up to 24 alphanumeric characters. • You specify a LEVEL statement once in a CONSTRUCT command.
LABEL <field>	<p>Specifies the external source fields containing the labels for the members of the corresponding attribute. You must separate all field names with a comma. Use a question mark (?) instead of a field name if no label exists for a particular attribute.</p> <p>Note: Labels can be up to 250 alphanumeric characters.</p> <p>Note: The label field cannot contain labels that begin with an asterisk and then a blank space. Labels can begin with other characters though. For example, the field cannot use a label * Massachusetts but it can use the label ~ Massachusetts.</p>
BY <attributeset>	If an attribute containing a hierarchy is associated with this dimension, <attributeset> specifies the name of the attribute.
BY <dimension>	Name of an existing dimension that the hierarchical attribute will be associated with.
SINGLE	Disallows multiple parents and can be used when you want to be alerted that your data contains relationships defining multiple parents. SINGLE generates an error message and stops the procedure from completing as soon as it detects a record which will cause a member to have multiple parents.
NOINPUT	Application Server does not read the data file.
KEY BOTH	Creates an index based on the dimension member names and labels. Keying a dimension allows quick selection of its members when selected by a member name or label.
ALLOCATE	Defines the maximum number of input and output members in a dimension. If you later add members to the dimension, the database is not reorganized.
ARRAYSIZE <n>	<p>Specifies a number between 1 and 32,768 for the size of the array you want to fetch from the RDBMS during the construct process. If you do not specify an array size, rows are fetched in batches of 100.</p> <p>Note: Some drivers do not support array fetching, such as the Microsoft ODBC driver for MS Access. In those cases, the array size will have a value of 1, and Link will only run single row fetches.</p> <p>Note: If you do not specify the ARRAYSIZE keyword on the command line or you do not add an ARRAYSIZE=n parameter in the <i>[linkid]</i> section of the LSDAL.INI file, the default array size value is 100. If you add the ARRAYSIZE parameter to LSDAL.INI, that value will be used instead of the default value. If you use the ARRAYSIZE keyword on the CONSTRUCT command, the CONSTRUCT ARRAYSIZE command will override all default values and ARRAYSIZE parameter settings in LSDAL.INI.</p> <p>Tip: The best array size may be different on different systems and networks, so you should experiment with array size numbers until you find the optimal value.</p>
REPLACE CALCULATIONS	Replaces the consolidation rules with new ones derived from the external file. By default, Application Server preserves any consolidation relationships that exist in the dimension and adds new rules based on the external file. Omitting the

CONSTRUCT ATTRIBUTE

	REPLACE CALCULATIONS keyword can result in members with multiple parents.
OVERWRITE	Rebuilds an existing dimension based on the information in the external data source. Rewrites consolidation rules, and deletes members that are no longer part of the external source from the dimension. Use of the OVERWRITE keyword generally requires a reconsolidation of the model.
REPLACE CLASSES	Removes any existing classes and uses only those defined in the external source. The default is to add members from the external source with the existing members.
SINGLE	Specify the SINGLE keyword to disallow multiple parents. If Application Server finds a record that would cause a member to have multiple parents, the procedure is terminated and an error message is displayed such as "Inconsistent Definition for (<member>) in Level (<level>)".
APEX	Indicates that instead of summing up the members into the last stated LEVEL on the construct statement, you do not want a summarized result. If you have multiple hierarchies in a constructed dimension, you can use APEX for each hierarchy. The first hierarchy gets a result and the other hierarchies get another OUTPUT summing the members of the last level in each hierarchy. For example, APEX TEST indicates that you want an apex (either RESULT or OUTPUT, depending on whether it is the first or subsequent hierarchies) called TEST. If you do not specify a name, the default name would be, for example, TOTAL_XXXX, where XXXX is the name of the last LEVEL in this hierarchy.
NOAPEX	Indicates you do not want an apex for this hierarchy.

6.40 CONSTRUCT ATTRIBUTE

Use

CONSTRUCT ATTRIBUTE constructs an attribute from an external source.

The CONSTRUCT ATTRIBUTE command produces an attribute set containing attribute names and members.

Note: If the attribute information exists in multiple files, you must use a CONSTRUCT ATTRIBUTE command for each source to construct multiple attribute sets that refer to the same dimension.

The template procedure

This command creates a template procedure with the same name as the attribute set if no sets by that name exist. You can use this procedure to populate the attributes with data. CONSTRUCT ATTRIBUTE creates this procedure if the set that contains the attribute information also contains the related structural information. For example, you could have one set that describes all your products, and another set that contains a product ID such as an SKU number, and attribute information like color, size, flavor, and so on.

If you had read the data from this set to construct the attribute descriptions, then you could enter JOB attribute_set_name to populate the created attribute variables. If your data were somewhere else, you could still do this by specifying a different data source and then running this procedure. This saves you from having to determine your across and down statements before populating your attributes.

Syntax

```
CONSTRUCT ATTRIBUTE <setname> BY <dimension>
  [VARIABLES <field> [...<field> ]]
  [LABEL <field> [...<field> ]]
  [CLASS <field> [MEMBER <field> ]]
  [HIERARCHY<hierarchy_name> [..., <hierarchy_name>]]
  [PREFACE "<keyword_string> [LEVEL [NAME '<level_name>'] [<keyword_string> ] " ]
```

Parameter	Description
<setname>	Name of the attribute set to create. You might want to use the same name as the dimension that this attribute will be associated with. If the attribute set will contain only one attribute, you might want to name the set the same name as the attribute.
BY <dimension>	Name of the dimension that the attributes will be associated with. Note: You can construct multiple attribute sets referring to the same dimension.
VARIABLES <field>	When constructing a simple attribute, <field> specifies the data source fields to use when building the attributes. You must separate all fields with commas. The field is the attribute name, such as size or color. Note: The field's values become attribute member names when constructed; member names can be up to 24 alphanumeric characters.
LABEL <field>	Specifies the external source fields containing the labels for the members of the corresponding attribute. You must separate all field names with a comma. Use a question mark (?) instead of a field name if no label exists for a particular attribute. Note: Labels can be up to 250 alphanumeric characters.
CLASS <field>	Identifies the fields to use when building the classes in the dimension, where <field> is the name of the external source field containing the class name(s). You can specify up to five CLASS clauses.
MEMBER <field>	Identifies a field, other than the input member field, to use when building classes in the dimension, where <field> is the name of the external source field containing the members of a class. If you do not specify a field, Application Server uses the members in the first level (input members).
HIERARCHY <hierarchy_name>	When constructing a hierarchical attribute, <hierarchy_name> is the name of the dimension created by the CONSTRUCT <hierarchy_name> command that contains an attribute's hierarchies. If you construct multiple hierarchical attributes, specify each one separated by a comma.
PREFACE "<keyword_string>"	Prefaces the resultant attribute set with the specified keyword(s). You must enclose the keyword string in quotation marks (' '). This option allows you to put anything in an attribute set. For example: CONSTRUCT ATTRIBUTE COLOR BY PRODUCT VARIABLE COLOR PREFACE "RESULT LEVEL" constructs an attribute set containing the RESULT LEVEL keyword.
LEVEL	Specifies a level name for the input level of the attribute so that the input level name is different from the attribute name. Use LEVEL NAME '<level_name>' when an attribute has the same name as one of its members and you want to avoid problems with duplication of names. If the attribute is named differently from its members, you do not have to use the LEVEL NAME '<level_name>' keyword.
NAME '<level_name>'	Specifies a name for the input level of the attribute. By adding a level name in situations where the attribute shares the same name as one of its members, you prevent duplicate dimension or attribute level or class names. You will get an error message if there is any conflict between a level name or class name and a member name or if a level name is duplicated in a multiple hierarchy dimension.
<keyword_string>	Allows you to add keywords to the LEVEL keyword. You must enclose the keyword string in quotation marks (' '). This option allows you to put anything in an attribute set.

Example 1: CONSTRUCT

...Create a dimension, Stores, containing three hierarchies, States, Regions, and Ownership:

CONSTRUCT ATTRIBUTE

ACCESS EXTERNAL

USE datafile

BEGIN

DESCRIPTION FREE TAB

store text 5

address text 32

salesregion text 5

saleslabel text 10

state text 2

georegion text 9

ownertype text 12

hours text 6

END

BEGIN

CONSTRUCT Stores

HIERARCHY States LEVEL store, state, georegion APEX geog LABEL address, ?,?

HIERARCHY Regions LEVEL store, salesregion LABEL address, saleslabel NOAPEX

HIERARCHY Ownership LEVEL store, ownertype APEX total_owner LABELaddress,?

CLASS hours NOTOTALS

PREFACE "KEY BOTH ALLOCATE 100,50,1"

END

END

COMPILE DIMENSION Stores

Example 2

...This procedure creates a hierarchy:

PROCEDURE setname

ACCESS subsystem

...statement(s) to select/describe the source

CONSTRUCT hierarchy_name1 LEVEL field, field LABEL field, field PREFACE "BY dimension"

CONSTRUCT hierarchy_name2 LEVEL field, field, field LABEL field, ?, ? PREFACE "BY dimension"

END

COMPILE DIMENSION

...Sample resulting set of <hierarchy_name1>:

BY dimension

INPUT

'member'

,'member ...'

OUTPUT

'member'

,'member'...

RESULT

TOTAL_hierarchy_name 'TOTAL hierarchy_name'

LEVEL

field

,field...

consolidation statements

Example 3

...This example adds the By Product "Custom *n*" to attributes on construct:

CONSTRUCT attribute_set PREFACE "Custom 2"

Example: CONSTRUCT ATTRIBUTE

...This procedure constructs a simple attribute and a hierarchical attribute:

PROCEDURE setname

ACCESS subsystem

...statement(s) to select/describe the source

BEGIN

CONSTRUCT ATTRIBUTE attributeset BY dimension VARIABLE field, ..., field

HIERARCHY hierarchy_name, hierarchy_name

END

END

COMPILE ATTRIBUTE attributeset

Example: Adding a class

...Construct an attribute set called Prodatt because it is associated with the Product dimension. The

... attribute is Color, and has two classes of colors, Pastels and Primary colors.

ACCESS LSLINK

CONNECT dbase

SELECT * from Product

CONSTRUCT ATTRIBUTE Prodatt BY Product VARIABLE color CLASS Shades

END

COMPILE ATTRIBUTE Prodatt

Constructing the dimensions

Note: Use these procedures as syntax examples - they will not work with the demonstration databases in your installation.

This procedure constructs and compiles the Drinks and Region dimensions:

...Construct the Region dimension

ACCESS LSLINK

CONNECT dbase

SELECT city, state FROM Drinks

CONSTRUCT region LEVEL city, state

...Construct the Drinks dimension

CONSTRUCT ATTRIBUTE

SELECT drinks **FROM** Drinks

CONSTRUCT Drinks **LEVEL** Drinks

END

COMPILE DIMENSION Region

COMPILE DIMENSION Drinks

The following is an example of the dimension set for the Region dimension. Application Server consolidates two input members into a Result member.

INPUT

COLA

,SPRITZ

RESULT

TOTAL_DRINKS 'TOTAL DRINKS'

LEVEL

DRINKS

TOTAL_DRINKS = INPUTS

The following is an example of the dimension set for the Region dimension. Application Server consolidates two input members into an output level. Application Server consolidates the output level into a result.

INPUT

PHOENIX

,BOSTON

OUTPUT

AZ

,MA

RESULT

TOTAL_REGION 'TOTAL REGION'

LEVEL

CITY

,STATE

AZ = PHOENIX

MA = BOSTON

TOTAL_REGION = SUM AZ, MA

Constructing the attributes

Note: Use these procedures as a syntax examples - they will not work with the demonstration databases in your installation.

This procedure constructs the attribute set, which contains a simple attribute and a hierarchical attribute. When the attribute set is compiled, the attribute variables pkg_type and size are also created.

ACCESS LSLINK

CONNECT dbase

SELECT Drinks, size, package, pkg_type **FROM** drinks

CONSTRUCT ATTRIBUTE attset BY drinks VARIABLE pkg_type HIERARCHY size

END

COMPILE ATTRIBUTE attset

The following is an example of the attribute set, attset. It shows that Application Server has constructed two attributes: the simple attribute Pkg_type and the hierarchical attribute Size.

ATTRIBUTE attset

BY DRINKS

VARIABLE PKG_TYPE

BOTTLE

,CAN

VARIABLE SIZE HIERARCHY

This shows the output of a SHOW Size command:

SIZE

Inputs

48 OZ. 32 OZ. 12 OZ. 8 OZ.

Outputs

Junior Giant

Result

TOTAL Size

4 Inputs 2 Outputs 1 Result

This shows the results of a SHOW pkg_type command:

PKG_TYPE

Inputs

BOTTLE CAN

2 Inputs

6.41 COPY

Use

You use COPY to copy sets into, out of, or within Application Server databases (EXTERNAL) or client (local) files. You can also use this command to copy from EXTERNAL to EXTERNAL.

Syntax

COPY [<settype>] <fromset>[;<database>] <to>[;<database>] [OVERWRITE] [RESOLVE]

COPY [<settype>] <fromset>[LOCAL|EXTERNAL] <to>[;<database>] [OVERWRITE] [RESOLVE]

COPY [<settype>] <fromset>[LOCAL|EXTERNAL] <to>[LOCAL|EXTERNAL] [OVERWRITE] [RESOLVE]

COPY [<settype>] <fromset>[;<database>] <to>[LOCAL|EXTERNAL] [OVERWRITE] [RESOLVE]

COPY [<settype>] TERMINAL <to>[;<database>]

Parameter	Description
<settype>	Type of set: dimension, document, logic set, procedure, report, synonym, time, or binary set types. If you do not specify a set type, Application Server will copy the first set it finds with the specified name.

CREATE Variable

<fromset>	Name of the set to copy.
<toaset>	Name of the output set.
<database>	Name of the database where the set is located. If you do not specify a database name, Application Server uses your Use database.
EXTERNAL	Indicates the set is in a text file that is not in an Application Server database or on the client machine. You can specify a full pathname.
RESOLVE	Performs any necessary control variable substitutions during the copy.
OVERWRITE	Overwrites a set with the same name.
LOCAL	Indicates that the text file is on the client machine (local).
TERMINAL	From within a batch job, can be used in place of <fromset>.

Examples: COPY

...Copy the dimension Product to the dimension Prod within the current Use database:

COPY dimension Product Prod

...This example copies the logic set Cashflow in the Bank database to the logic set cashflow in the

... Corp database. Both Corp and Bank must be attached.

COPY logic cashflow;bank cashflow;corp

...This example copies the procedure Setqtr to the external file Setupq:

COPY procedure setqtr setupq;EXTERNAL

...Copy the local DOS or Windows file c:\lsserver\myfile to a UNIX path /lsserver/MYFILE:

COPY procedure 'c:\lsserver\myfile' /lsserver/MYFILE

COPY procedure 'c:\lsserver\myfile' ;local /lsserver/MYFILE;external

...This example displays the settype Proc and copies it to the procedure

...Document, when it is part of a procedure and executed with the JOB command:

COPY procedure TERMINAL document

6.42 CREATE Variable

Use

Creates one or more normal variables, virtual variables, or text variables.

Syntax for normal variables

```
CREATE [<periodicity>] <variable> ['<label>']
    [, ... <variable> ['<label>']]
    [INTO <remotedb>]
    [TEMPORARY]
    {LIKE <like_variable>}
    {BY <dimensions> [INCLUSIVE | EXCLUSIVE [ZEROS]] [FIRST | LAST | AVERAGE | SUM |
    WEIGHTED <weighted_variable> | MIN | MAX] }
    [RATE]
    [UNITS]
    [SPARSE | DENSE]
    [INTEGRAL]
    [BYTES <n>]
    [EXPENSE] }
```

Syntax for virtual variables

```
CREATE <variable> ['<label>'] [BY <dimensions>]
    {AS {<expression> / COUNT OF <variable> / DISTINCT COUNT BY <dimension> [, <dimension>...]} OF
    <variable> } / LOGIC <logic> }
```

Notes:

Virtual measures do not support periodicity-based conditions, for example, WHEN YEAR IS XXX.

Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.

Syntax for text variables

```
CREATE [<periodicity>] TEXT <variable> ['<label>']
    [...<variable> ['<label>']]
    [TEMPORARY]
    [LIKE <like_variable> | BY <dimensions>]
```

Parameter	Description
<periodicity>	<p>Applies one of the following periodicities to the variable(s): constant, yearly, semiannually, quarterly, bimonthly, monthly, lunar, biweekly, weekly, daily, or hourly. When creating numeric variables, monthly is the default; when creating text variables, constant is the default. A constant periodicity assigns one value across all periods to the variable.</p> <p>Note: If you are creating biweekly measures, then you must create a custom calendar to support them. See the USER STARTING ENDING keywords in the SET FISCAL command.</p>
<variable>	<p>Name of the variable to create, up to 96 bytes starting with a letter. You must enclose names with embedded spaces and names that contain special characters with single quotation marks (' '). You can use zeros, digits, periods, or underscores. Underscores print as blanks on output. When creating normal or temporary variables, you can enter multiple variable names and optional labels separated by commas.</p> <p>Note: Variable names and labels should not include colon (:) characters.</p>
'<label>'	<p>Name of the label to assign to the preceding variable, enclosed in single quotation marks (' '). When you assign a label, it prints instead of the name.</p> <p>Notes:</p> <ul style="list-style-type: none"> If you create labels for variables, the first 24 characters of each label must be unique. You cannot specify a label that begins with an asterisk and then a blank space. You can use other characters though. For example, you cannot specify '* sales' but you can specify the label '~ sales'.
INTO <remotedb>	Creates a variable that is stored in another, existing, database, as specified by <remotedb>.
TEMPORARY	<p>Creates a temporary variable that is stored in the Work database, but not in the Use database. It is useful to create a temporary variable when you attach to a database in read-only mode and you want to create, calculate, and then remove a variable.</p> <p>You can calculate, consolidate, and select a temporary variable as you can a permanent variable. You can display information about temporary variables using the SHOW VARIABLES command.</p> <p>Application Server saves a temporary variable in the Work database when you exit. It is saved in the Use database when you save data and then exit clear. You can permanently remove a temporary variable using the REMOVE variable TEMPORARY command.</p>
LIKE <like_variable>	Creates variables with the same characteristics (dimensions, periodicity, and so on) as another variable, where <like_variable> is the name of the variable whose characteristics you want to copy.
BY <dimensions>	Creates variables dimensioned by the specified dimensions, where <dimensions> is one or more dimension names that the variable is dimensioned by, separated by commas.

CREATE Variable

FIRST	Uses the first value of the variable in the time series when Application Server consolidates the variable over time.
LAST	Uses the last value of the variable in the time series when Application Server consolidates the variable over time (the default for text variables).
INCLUSIVE EXCLUSIVE [ZEROS]	<p>INCLUSIVE means that missing values are included in the conversion.</p> <p>EXCLUSIVE means that missing values are excluded from the conversion.</p> <p>ZEROS treats zeros as missing values. Use this to treat data from BW properly by excluding zero values from the conversion.</p>
AVERAGE	Uses the average value of the variable in the time series when Application Server consolidates the variable over time (the default for rate variables). An INCLUSIVE AVERAGE variable will count a missing time period as zero. An EXCLUSIVE AVERAGE variable will ignore missing values when calculating the average value for the variable.
SUM	Uses the sum of the values in the time series when Application Server consolidates the variable over time (the default for all variables, except rate and text).
WEIGHTED <weighted_variable>	Uses the weighted average of the variable. For example, price might be weighted by units sold. The quarterly price for monthly data is the sum of price multiplied by the number of units sold during the months, divided by the sum of the units sold for those same three months.
MIN	Uses the minimum value of the variable in the time series when Application Server consolidates the variable over time. For example if you view a monthly measure quarterly and the measure is MIN, you see the smallest monthly value as the quarterly value.
MAX	Uses the maximum value of the variable in the time series when Application Server consolidates the variable over time.
RATE	Defines the variable as a rate variable. Application Server does not consolidate rate variables for output members.
UNITS	Indicates that values derived when the variable is used at another periodicity are rounded to the nearest integer.
SPARSE	<p>Indicates the variable contains many consecutive, identical values for the time series or many consecutive MISSING values (the case where only a few of the periods have data values in them). Application Server compresses this data and gains significant disk storage savings with only a small increase in access time. This is the default setting. If you do not specify SPARSE or DENSE in the CREATE command, the variable will be created as SPARSE unless you have SET SPARSE OFF. A SPARSE variable will store each series using the most space efficient method for that series.</p> <p>A SPARSE variable ignores the MULTIPLES keyword used on CREATE DATABASE. The series will be stored using as little space as possible, with no extra padding to allow for additional periods of data to be added to it later. If you do add additional periods of data later, for example say you load data for the first two quarters and then some time later load data for the third quarter, then this will mean that the series will have to be rewritten because the amount of storage may change. If you do this for many series then this may lead to the database becoming disorganized because many series have had to be moved around database blocks. In that situation you may be better off creating DENSE measures, which have extra padding in them to house the additional periods. If you do not add additional periods of data to existing series, for example if to recreate you database and reload all the data every time you should always be better off creating SPARSE measures.</p> <p>Note: After creating a variable, you can use the SHOW command with the SPARSITY keyword to check whether the variable actually has sparse or dense series. If you want to change the way a variable is stored from sparse to dense or from dense to sparse, you can dump the database, use the SET SPARSE ON OFF command to turn on or off sparsity in the dump file, and then load the dump</p>

	file. The variables will be created according to the SET SPARSE setting when the dump file is loaded.
DENSE	<p>Indicates the variable is dense.</p> <p>Note: SPARSE is a better method for creating measures when databases are removed, recreated and populated from scratch each time and no additional data for extra periods is added to the series in that database. DENSE may be a better method for creating measures when additional data for extra periods is added because DENSE measures honor the MULTIPLES keyword, while SPARSE ones do not. This allows extra space in series to house the extra periods.</p>
INTEGRAL	<p>Indicates that Application Server stores data as integers instead of double precision. 1, 2, or 4 bytes per value are used instead of 8.</p> <p>The following values will be accepted into integral variables:</p> <p>integral bytes 1 stores data between 0 and 254</p> <p>integral bytes 2 stores data between 0 and 32,766</p> <p>integral bytes 4 stores data between 0 and 2,147,483,647</p>
BYTES <n>	Number of bytes to use per data value. <n> can be 1, 2, or 4 with INTEGRAL, or 4 without it (single precision). The default number of bytes per data value is 8 (double precision), which allows 15 digits of precision. Single precision allows six digits of precision. Thus, a million dollar sales figure can be in error by as much as 10 dollars. You can reduce database size by specifying single precision for variables that do not require accuracy to the penny.
TEXT	<p>Indicates the variable contains characters. If you do not specify another periodicity, the default periodicity for text variables is constant. The maximum number of characters for any value in a text variable is 255.</p> <p>Note: Although a variable's value may contain up to 255 characters, the SET VARIABLE WIDTH, SET DEFAULT WIDTH, and SET-WIDTH-ENDSET commands in the Report editor allow you to specify a width up to 50 characters.</p>
EXPENSE	Specifies that the variable as an expense variable. This enables you to use commands such as WHEN <variable> IS EXPENSE.
DESCRIPTION <description>	Literal text that is displayed next to the variable when you enter SHOW VARIABLES.
<expression>	<p>Specifies the expression to use to calculate the normal virtual variable. You can specify any expression, for example CREATE MARGIN AS Sales-Costs, except one containing time calculations.</p> <p>Expressions cannot be longer than 999 bytes long.</p>
LOGIC <logic>	Specifies a logic to use to calculate the virtual variable.
COUNT OF variable	Creates a virtual variable that is the count of the number of individual time series that make up a particular view.
<i>DISTINCT COUNT BY</i> <dimension> [, <dimension>...] OF <variable>	<p>Creates a virtual variable that is a distinct count of the number of individual time series for that dimension.</p> <p>Note: Alternatively, if you want to display the distinct count of a variable by certain dimensions, you can simply select the variable and specify the DISTINCT COUNT BY clause in the SELECT variable command. The benefit to creating a virtual variable for the distinct count is that you can provide a label, a name, and also see the variable in the Selector and Viewer dialog boxes. See the SELECT <variable> command for information about the alternative method.</p>

Example 1

...This example creates a yearly variable YSales with the label Yearly Sales.

```
CREATE yearly YSales 'Yearly Sales' by Product, Customer,
Region
```

...This example creates a text variable Contact.Name dimensioned by Customer:

```
CREATE TEXT Contact.Name BY Customer
```

CREATE Variable

...This example creates a monthly rate variable Price dimensioned by Product. The data is expected not to change much from month to month. For example, the price of a product might be \$1.45 in January and \$1.65 for the next 14 months.

CREATE Price BY Product RATE SPARSE

Example 2

...Create a virtual variable named Margin% and dimensions it by Product and Channel.

CREATE 'Margin%' BY Product, Channel as ((Sales - Costs)/Sales) * 100

Creating the variables, reading data into them, and consolidating

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

This procedure creates the sales, broker_com, and cases variables and reads data into the variables:

...Create the variables

CREATE sales, broker_com, cases BY product, customer

...Select the items and set up the view

ACROSS var DOWN product, customer, time

SELECT var BY product, customer

SELECT product input

SELECT customer input

SET PERIOD 1998

...Read data into the variable

ACCESS LSLINK

CONNECT dbase

SELECT * FROM transact

STATUS

PEEK ONLY 10

LSS CREATE product = code

LSS CREATE customer = account_co

LSS CREATE time = SUBSTR(date,1,2) + "/" + SUBSTR(date,3,2)

SELECT * FROM transact

READ ADD

END

...Consolidate

SELECT Product

SELECT Customer

SELECT var BY product, customer

ROLLUP sales

ADD EVERYBODY

END

CONSOLIDATE

The result of a SHOW VARIABLES command looks like this:

PRODUCT:

Variable	Type	Items	Date Range
FLAVOR	AT [8]	CO	
PACK_CNT	AT [3]	CO	

CUSTOMER, PRODUCT:

Variable	Type	Items	Date Range
BROKER_COM	NU	12 MO	Jan 98 - Dec 98
CASES	NU	12 MO	Jan 98 - Dec 98
SALES	NU	12 MO	Jan 98 - Dec 98

6.43 CREATE (Access) or LSS CREATE

Use

CREATE is an Access command that can be used to manipulate text fields from external data records to create new fields. You can concatenate two or more fields, combine fields with literal strings, or use a substring expression on a field.

Note: You cannot use the CREATE command on fields that represent measure data. If you wish to rename a numeric field, either use field aliases within the SQL statement or use Read Logic <Logic SetName>.

Syntax

CREATE <field> = <piece1> + <piece2> + ...

Note: Use the following syntax in ACCESS LSLINK so as not to confuse the command with the SQL CREATE command.

LSS CREATE <field> = <piece1> <piece2> + ...

Parameter	Description
<field>	Name of the field to create.
<piece1>	A quoted literal string, field name, or substring expression. A substring expression has the format: SUBSTR(<field>, <position>, length) <field> Name of a new or existing field. <position> <position> in the field to start. <length> Number of characters to include in the field.

Note: CREATE cannot be used when reading data into attribute sets. The only method of renaming fields for use with Attributes is to use SQL field aliasing in the SQL statement. For example, the following would produce an error:

```

SELECT VARIABLE FLAVOUR
SELECT PRODUCT INPUT
ACROSS VARIABLE DOWN TIME, PRODUCT
ACCESS EXTERNAL
CONNECT DBASE
SELECT SKU, FLAVOR FROM DBFILE.DBF
LSS CREATE FLAVOUR = FLAVOR

```

CREATE DATABASE (Supervisor)

```

READ SAVE 1000
END

```

but this method will work correctly:

```

SELECT VARIABLE FLAVOUR
SELECT PRODUCT INPUT
ACROSS VARIABLE DOWN TIME, PRODUCT
ACCESS EXTERNAL
CONNECT DBASE
SELECT SKU, FLAVOR FLAVOUR FROM DBFILE.DBF
READ SAVE 1000
END

```

Examples

...If your field Month does not contain the year, but only the values 1 to 12,

...you can append the year with a statement like the following:

```
CREATE Date = Month + '99'
```

...If two fields contain the date, the month as the 3rd and 4th digits in a field called Paymentdate, and

... the year in a field called Year, you can append the year with a statement like the following:

```
CREATE Date = SUBSTR(Paymentdate, 3, 2) + Year
```

6.44 CREATE DATABASE (Supervisor)

Use

A Supervisor command that creates a new database.

Syntax

```

CREATE DATABASE <database>
  [ EXHIBIT | NOEXHIBIT ]
  [ BLOCKS <blocks> ]
  [ BLKSIZE <blksize> ]
  [ PARTITIONS <partitions> [SIZES <size> [...], <size>] ]
  [ PREFORMAT ]
  { OBSERVATIONS <obs> | MEMBERS <members> }
  [ OVERWRITE ]
  [ DEFER ]
  [ PROTECTION <protection> ]
  [ UPDATE <update> ]
  [ READ <read> ]
  [ SORTKEYS {ALPHABETIC | NMEMBERS } ]
  [ MULTIPLE <multiple> [INPUT] ]
  [ GIGABYTES 16 | 32 | 64 | 128 | 256 | 512 ]
  [ USAGE [EXCLUSIVE | READ | SHARED] [RETAIN] ]
  [ VARIABLES <var> ]

```

Parameter	Description
<database>	Name of the database to create. If you are creating a database without partitions, <database> can be up to eight characters long. If you are creating a database with partitions, <database> can be up to six characters long (five characters long if you are creating 100 or more partitions). Do not use embedded underscores in the database name.

EXHIBIT	Specifies that the database details will always be displayed by the EXHIBIT DATABASES command. This is the default value.
NOEXHIBIT	Specifies that the database details will only be displayed by the EXHIBIT DATABASES command if you use the command with the HIDDEN keyword.
BLOCKS <blocks>	Specifies the number of blocks of physical disk space for the database. The default is 200. Application Server does not preallocate blocks; it uses them on an as-needed basis. Therefore, you should check the availability of space because it is possible to specify a block size and then find that at the time Application Server actually uses the blocks, they are not available.
BLKSIZE <blksize>	Specifies the number of bytes per block: 512, 1K, 2K, 4K, 8K, 16K, 32K, 64K, or 128K. 16K is the default, and is recommended because it causes less I/O. The initial database potential capacity is <blocks> x <blksize>.
PARTITIONS <partitions>	Number of database partitions to create, up to 256. Specify the PARTITIONS keyword after the BLKSIZE keyword. See Creating a dimensional model with multiple partitions for more information.
SIZES <size>	One or more numbers that identify the size of each new partition in blocks. You must separate each size with a comma. You can specify <size> as the number of blocks (for example, 20,000), as a multiple of 1,000 with a K suffix (for example, 20K), or as a multiple of 1,000,000 with an M suffix (for example, 1M). By default, Application Server divides the partition sizes evenly based on the number of blocks. If you specify fewer sizes than the number of partitions, Application Server divides the remaining blocks equally among the remaining partitions.
PREFORMAT	Preformats the database up to the database size as specified by the BLOCKS keyword. Application Server writes to all the blocks to ensure that the space is preallocated.
OBSERVATIONS <obs>	<p>Specifies the maximum number of observations a time series can contain. To determine this number, multiply the periodicity of each variable by its date range. For example, if the variable Sales will contain monthly observations from January 1999 to December 2000 inclusive, you will have two years of monthly data, making a total of 24 observations.</p> <p>In addition, the <obs> parameter determines the maximum number of columns that can be displayed. For example, if you have 52 weeks of data for two variables, and you want to display variables and time across for the whole 52 weeks, you must ensure that the <obs> value is set to at least 104 (equal to 52 observations * two variables).</p> <p>Note: The maximum number of observations for a database is 32,000.</p>
MEMBERS <members>	Specifies the number of members in the largest dimensions. See Using the MEMBERS keyword for more information.
OVERWRITE	Specifies that Application Server should overwrite any existing file with the same name.
DEFER	Use DEFER on systems where write access to the database storage location is restricted to a specific user.
PROTECTION <protection>	<p>Name of the protection key that is required to remove the database record from one MASTERDB to another. You can specify a protection key with up to 16 characters.</p>
READ <read>	One or more read access keys separated by commas. Each read access key is an alphanumeric name up to 16 characters. You can set both a READ key and an UPDATE key on a database. Use the Supervisor REMOVE command to remove a read or update access key.
SORTKEYS { ALPHABETIC NMEMBERS }	<p>Determines the criteria used for the order that dimensions are referenced in internal Application Server database key structures. ALPHABETIC orders the dimensions alphabetically and NMEMBERS orders the dimensions based on the number of members in each dimension. To optimize performance, it is recommended that NMEMBERS be used. However, ALPHABETIC remains the default for backward compatibility reasons.</p> <p>Notes:</p>

CREATE DATABASE (Supervisor)

SORTKEYS cannot be abbreviated; you must type it in full.

When using virtual cube models, all of the Application Server databases that make up the virtual cube model must be of the same **SORTKEYS** type. Also, if you use **SELECT VARIABLE <variable> FROM <database>** to select variables in an attached model, the **USE** and **ATTACH** models must have the same key sorting method.

In order to change the **SORTKEYS** type for a model, you must recreate it. It is possible, however, to load a dump file from a **SORTKEYS ALPHABETIC** model into a new database created with **SORTKEYS NMEMBERS**. The data keys will be converted at load time. All procedures that issue **ROLLUP** commands with specific quadrant numbers should be validated when converting a model from one **SORTKEYS** type to another since quadrant numbers are likely to change.

UPDATE <update> One or more update access keys separated by commas. Each update access key is an alphanumeric name up to 16 characters.

MULTIPLE <multiple> Specifies the number of values that can be added to a time series before the record size is increased, where **<multiple>** is a numeric value between 1 and 255. The default is 6, which means that 6 values can be added to the time series. When the 7th value is added, Application Server allocates space for another 6 values. To apply the **MULTIPLE** setting to input time series only (and not to consolidated time series), add the keyword **INPUT**, for example:

MULTIPLE 10 INPUT

GIGABYTES 16 | 32 | 64 | 128 | 256 | 512

The maximum size of the database. You should set this parameter to the anticipated size of your database, not larger. A maximum database size of 32 GB is currently tested and supported.

USAGE [EXCLUSIVE | READ | SHARED] [RETAIN]

Indicates the access mode when a user enters a **USE** or **ATTACH** command without specifying a *usage* parameter. You can specify **READ** for read access, **SHARED** for shared access, or **EXCLUSIVE** for exclusive access (the default). In addition, you can specify the **RETAIN** keyword to keep the current database attached.

VARIABLES <var> The maximum number of variables the database can contain. The default value is 1000, and the maximum is 10,000.

Example

...This example creates a database **BUDGET**:

CREATE DATABASE budget BLOCKS 700 OBSERVATIONS 100

...This example creates a 500-block database **PAYABLES**:

CREATE DATABASE payables BLOCKS 500 OBSERVATIONS 400 UPDATE payable

...The following example creates a database called **ACCNT** with 400,000 blocks in a

...50,000-block partition and a 350,000-block partition:

CREATE DATA Accnt BLOCK 400000 BLKSIZE 8K OBS 500 PARTITIONS 2 SIZES 50K, 350K

...The output of this command would be:

Database: **ACCNT**

Maxaccess: **UPDATE** Disposition: **ENABLED**

Current State: **AVAILABLE** Protection Key: **None**

Last user: Last access:

Blksize: **8192** Observations **500**

Maxblocks: **400000** Number Free: **399990**

Database **ACCNT** Partition Information.

Number of Partitions: 3
 Blocksize: 8192
 Date Last Opened for Update: Thu Feb 9 15:32:59 1999
 Date Last Closed After Update: Thu Feb 9 15:32:59 1999

Partition	Size	Name
1	50000	ACCNT01
2	350000	ACCNT02

...The following example creates a database called DEMO with 600,000 blocks in three
 ...partitions of 200,000 blocks each. They are named DEMO01, DEMO02, and DEMO03.

CREATE DATA DEMO BLOCK 600000 PARTITIONS 3

...The following example creates a database called ACCNT with 10,000 blocks, with
 ...each block 8K in size, and with the number of members in the largest dimensions
 ...not to exceed 36,000 members:

CREATE DATABASE ACCNT BLOCK 10000 BLKSIZE 8K MEMBERS 36000

...This example creates a database called ACCNT with 3 partitions 200K in size.

...Application Server writes to all the BLOCKS to ensure that the space is preallocated:

CREATE DATABASE ACCNT PARTITION 3 SIZE 200 PREFORMAT

...This example creates a database called ACCNT 12000K in size. Application Server
 ...writes to all the BLOCKS to ensure that the space is preallocated:

CREATE DATABASE ACCNT SIZE 12000 PREFORMAT

...This example creates a database called ACCNT with maximum size of 64GB:

CREATE DATABASE ACCNT GIGABYTE 64

6.45 CREATE Dimension User_Defined_Hierarchy

Use

CREATE <dimension> <user_defined_hierarchy> creates User-Defined Hierarchies containing the specified members of a dimension. You can create any number of User-Defined Hierarchies for a dimension, up to the number specified in the CUSTOM statement in the dimension set. You also use the CREATE <dimension> <user_defined_hierarchy> command to update User-Defined Hierarchy definitions.

Notes:

You specify the number of User-Defined Hierarchies for a dimension in the dimension set when you construct the dimension, or you manually add the CUSTOM statement to the dimension set.

When switching databases, all selections related to the first database, such as User-Defined Hierarchy creations, are discarded.

A User-Defined Hierarchy is created in the level above its highest members. For example, if all members of a User-Defined Hierarchy are input members, the User-Defined Hierarchy is created at the first output level.

Syntax

CREATE <dimension> [REPLACE] [NOCORRECTIONS] <user_defined_hierarchy> [<label>] = {
 SELECTED | <expression> }

CREATE Dimension User_Defined_Hierarchy

Parameter	Description
<dimension>	Name of the dimension whose User-Defined Hierarchies you want to define. The CUSTOM statement must exist in the first line of the Dimension editor.
REPLACE	Replaces the existing User-Defined Hierarchy definition with a new definition. If this User-Defined Hierarchy is used as a member of other User-Defined Hierarchies, the definitions will be updated in all instances. When you use the REPLACE keyword, it replaces the definition for the User-Defined Hierarchy and also updates the procedure that recreates the User-Defined Hierarchy. Note: If you want to save these changes for the next session, issue a SAVE CUSTOM dimension CHANGE command.
NOCORRECTIONS	Performs multiple counting when any descendants of the User-Defined Hierarchy causes multiple counting within the User-Defined Hierarchy.
.	Application Server will not search for multiple counting issues in dimensions compiled and all subsequent aggregations performed on those dimensions are aggregated exactly as defined in the hierarchy with no corrections. If you omit NOCORRECTIONS, any multiple counting within nested User-Defined Hierarchies or multiple counting arising from descendants of User-Defined Hierarchies will automatically be corrected in the aggregations performed at run time. For information, see <i>Multiple Counting in Dimensions</i> .
<user_defined_hierarchy>	Name of the User-Defined Hierarchy to define for this dimension.
<label>	Optional label for a User-Defined Hierarchy member. Must be in double quotation marks (" "). Use up to 250 characters.
SELECTED	Adds only selected members to this User-Defined Hierarchy.
<expression>	<member> ["<label>"] [+ <member> - <member>] [+ <member> - <member>] ... or <user_defined_hierarchy> [+ <user_defined_hierarchy> - <user_defined_hierarchy>] ... <member> Members to add or remove from the User-Defined Hierarchy. You can add and subtract members using the plus (+) and minus (-) signs. <label> Optional label for a User-Defined Hierarchy member. Must be in double quotation marks (" "). Use up to 250 characters. <user_defined_hierarchy> Name of an existing User-Defined Hierarchy to add or remove from the User-Defined Hierarchy. You can add and subtract User-Defined Hierarchies using the plus (+) and minus (-) signs.

Examples: CREATE dimension user_defined_hierarchy

..This example creates a user_defined_hierarchy based on the selected members of the dimension:

```
SELECT Product color_monitors, hard_drives
```

```
CREATE Product Hot_Products=SELECTED
```

...This example creates a user_defined_hierarchy called Hot_Products based on the specified

...members and a previously defined user_defined_hierarchy:

```
CREATE Product Hot_Products=color_monitors+hard_drives+my_products
```

...Create a user defined hierarchy called Hot, labeled as "Hot Products" based on specified members:

```
CREATE PRODUCT HOT "Hot Products" = CARS+VANS
```

6.46 CREATE GROUP (Supervisor)

Use

A Supervisor command that creates a User-Defined Hierarchy definition in MASTERDB. You can add users to the User-Defined Hierarchy and then reference the User-Defined Hierarchy in a Security procedure to restrict those users' access to specific dimensions.

Syntax

```
CREATE GROUP <user_defined_hierarchy>
```

Parameter	Description
<user_defined_hierarchy>	Name of the User-Defined Hierarchy to create. You can create any number of User-Defined Hierarchy.

Example

...The following example creates a User-Defined Hierarchy called Finance:

```
CREATE GROUP Finance
```

6.47 CREATE USER (Supervisor)

Use

CREATE USER is a Supervisor command that creates a new user record in MASTERDB.

Syntax

```
CREATE USER <user>
    [ USEDDB <usedb> ]
    [ USAGE [EXCLUSIVE | SHARED | READ] ]
    [ WORKDB <workdb> ]
    [ BLOCKS <blocks> ]
    [ BLKSIZE <blksize> ]
    [ BUFFERS <buffers> ]
    [ SECURITY [0 | 99] ]
    [ LOGIN [ON | ENABLED | DISABLED | PAUSED] ]
    [ EXPIRATION <date> ]
    [ PASSWORD <password> ]
    [ MAXLOGIN <lognum> ]
    [ ACCESS <key> ]
```

Parameter	Description
<user>	Specify a name for the new user. The user name must begin with an alphabetic character but the rest of the name can be alphanumeric. You can specify up to 96 bytes. Note: User names must not contain spaces or underscore characters (_).
USEDDB <usedb>	Specifies the database the user is attached to after logging in. If you omit this parameter, Application Server attempts to attach the user to a database with the same name as the user name.
USAGE [EXCLUSIVE SHARED READ]	Specifies the mode in which the default database is opened. You can specify READ for read access, SHARED for shared access, or EXCLUSIVE for exclusive access. This setting is used when logging in.
WORKDB <workdb>	Specifies the file name Application Server should use to create the Work database. You can specify an alphanumeric name of up to eight characters. Do not include a path; Application Server creates all Work databases in the home

CREATE VERSION

	directory. The default is DB<nnnn>. The user has read and update access to this database.
BLOCKS <blocks>	The number of blocks that should be allocated to the Work database. The minimum is 2000 blocks and the default is 2000 blocks.
BLKSIZE <blksize>	The size of blocks in the Work database in bytes. You can specify 512, 1K, 2K, 4K, 8K, or 16K. The default is 16K.
BUFFERS <buffers>	The number of buffers to use. The minimum is 20 and the default is 2000.
SECURITY [0 99]	The user's security level. Specify 0 for a normal user or 99 for a supervisor. The default is 0.
LOGIN [ON ENABLED DISABLED PAUSED]	Specifies the state of the users login. Specify ON, ENABLED, DISABLED, or PAUSED.
EXPIRATION <date>	The date on which the user's account expires. After this date, the user will be unable to access Application Server.
PASSWORD <password>	The user's password. Specify an alphanumeric name with up to 12 characters. The first character must be alphabetic, not numeric.
MAXLOGIN <lognum>	The number of times the user can log into Application Server concurrently. Specify a number between 1 and 32,000. The default is 255. You might want to allow a user multiple logins when many users within a User-Defined Hierarchy or department will use the same login ID to access Application Server.
ACCESS <key>	Specifies an access key that is required to access a database in either read or update mode.

Example

...This example creates the user Mel with default Use and Work databases:

```
CREATE USER Mel USE salesdb WORK wkmel
```

...This example creates the user Mel and sets an access key of "account1" for that user:

```
CREATE USER Mel ACCESS account1
```

6.48 CREATE VERSION

Use

CREATE VERSION duplicates the structure and data of variables in a separate copy.

Syntax

```
CREATE VERSION <version> [REPLICATE [DATA]]
```

Parameter	Description
<version>	Name of the version you want to create. The default version name is DEFAULT.
REPLICATE	Duplicates the structure of all variables in the version.
DATA	Duplicates the data of all variables in the version.

Example: CREATE VERSION

..This example duplicates the structure of all the existing variables in the version OPTIMISTIC:

```
CREATE VERSION optimistic REPLICATE
```

6.49 CSWITCH

Use

Use the CSWITCH command to clear the current user and Work database before you log into Application Server again as a different user. Or, you can also switch to a different Work database in

the same session while deleting the previous Work database and staying logged in as the same user.

Syntax

```
CSWITCH { <username> [PASSWORD <password>] | WORKDB [<name>] }
```

Parameter	Description
<username>	The user name you want to log in as.
PASSWORD	Indicates that the user name has a password.
<password>	The password for that user.
WORKDB	Clears all prior Work databases in the session and creates a new Work database named DBxxxxx while remaining logged in as the same user. Note: See the SWITCH command if you want to switch to a different Work database and save the previous Work database.
<name>	<p>Clears all prior Work databases in the session, and switches to the Work database name specified while remaining logged in as the same user. If the name already exists, it is used and the state is restored. If that Work database exists but is not a valid Work database, you get an error and rollback.</p> <p>If the Work database does not exist, a new one is created with the blocksize and blocks defined for the current user, that is, just as original Work database would have been created.</p> <p>Note: Some settings like the ACROSS and DOWN commands and SELECT statements are saved in a Work database. If you switch back to a previous Work database, you will get those settings, and as a result, your view may change.</p> <p>Tip: The WORKDB feature is helpful in applications that use User-Defined Hierarchies, such as the strategy management application. The strategy management application typically creates and clears User-Defined Hierarchies for each transaction. By switching between multiple Work databases, you can maintain the appropriate state and avoid the constant creating and clearing of the User-Defined Hierarchies.</p>

6.50 DECONSTRUCT

Use

DECONSTRUCT is an Access command that generates input data from a compiled dimension that can be used to later CONSTRUCT the same dimension. DECONSTRUCT will work with ACCESS EXTERNAL or ACCESS LSLINK.

Note: For ACCESS EXTERNAL if no DESCRIPTION is supplied, DECONSTRUCT will generate a default DESCRIPTION internally. For ACCESS LSLINK you must supply a SELECT statement from an existing table that you want the output to be inserted into.

Syntax

DECONSTRUCT [DIMENSION] <dimname> [LABEL SUFFIX <literal>] [NOATTRIBUTES] [NOSPACES] [PROCEDURE {<set> | <filename>}] [OVERWRITE]

Parameter	Description
DIMENSION	Option keyword.
<dimname>	Name of the dimension to deconstruct.
NOATTRIBUTES	Specifies that you do not want to have fields for any attributes of the dimension in the output file.
LABELSUFFIX <literal>	Specifies the suffix that you want applied to LEVEL fields to produce a LABEL field. For example in JUICE for dimension PRODUCT there are levels PRODUCT, BRAND and CATEGORY. The output file needs fields for LABELS corresponding to each level. By default it will generate fields called PRODUCTLABEL, BRANDLABEL, CATEGORYLABEL by appending a suffix "LABEL" to the level name. You can supply an alternative suffix to use that instead.
NOSPACES	Specifies that all the fields will have no spaces in them in the output file. Any spaces will be written as underscores in the output. This allows you to use the NOSPACES keyword in the DESCRIPTION statement in ACCESS EXTERNAL if you want to use the output file created by DECONSTRUCT as input to a subsequent CONSTRUCT command.
PROCEDURE {<set> <filename>}	Generates a procedure that will have all the statements to run a CONSTRUCT from the output of the DECONSTRUCT command. The procedure can be a procedure in PAS or an external text file. The procedure will include a PREFACE statement to generate any CUSTOM or ALLOCATE statements in the original dimension.
OVERWRITE	Specifies that you want to overwrite a system file with the same name, if there is one.

6.51 DEFINE SYNONYM (Access)

Use

DEFINE SYNONYM is an Access command that assigns a synonym to a dimension.

Syntax

DEFINE SYNONYM <dimension><synonym>

Parameter	Description
<dimension>	Name of the dimension, including Time and Variables, whose members have different names in an external data source.
<synonym>	Name of the synonym set that contains all the synonyms for a given dimension.

Example

...If the variables Sales and Expenses are known as Msales and Mcosts in
...an external data source, you can create a synonym set, VARSYM, that contains:

...Sales Msales

...Expenses Mcosts

...Then in the Access subsystem, enter the following statement:

DEFINE SYNONYM Variables Varsym

6.52 DESCRIPTION (Access External)

Syntax

```
DESCRIPTION [BINARY <length>] [FREE <separator>| FIXED] [MULTIPLE]
[DECIMALPOINT '<character>']
.
.
.
<field> [DATE] [<type> [UPPERCASE] [NOSPACES] ] <width> [<decimals>] [<offset>]
.
.
.
```

6.53 Description

DESCRIPTION is an Access External command that defines an external file format.

Parameter	Description
BINARY	Indicates the external file is a binary file.
<length>	Record length in the binary file. This value is an integer less than 32,768.
FREE	Indicates the external file is a free format file.
<separator>	A punctuation symbol (, ^ ; # / \ . : =), the word SPACE or TAB, an integer, or a hexadecimal number representing the value of the separator character in your character set.
FIXED	Indicates the external file is a fixed format file (the default).
MULTIPLE	Indicates logical records read are represented by more than one physical record; however, each logical record must begin a new physical record.
DECIMALPOINT	The decimal point character used in the external file.
<field>	Name of the field of up to 24 alphanumeric characters starting with a letter. You must enclose names with spaces in single quotation marks (' '). You would typically use names of dimensions or variables. For example, a field might be Time, Sales, Product, or Hour. You can specify more than one field statement. Note: You can define a maximum of 2048 fields.
DATE	An optional prefix to the date format type.
<type>	Type of data the field contains: BINARY Valid only with binary files. <date format> Can be: myy, ymd, yyym, dmyy, dmy, mdy, ydm, yydm, yym, my, or ym If the year is four digits (2000), the formats with yy are used. If the year is two digits (00), the formats with y are used. If the month is alphabetic (May 2000), the letter m is replaced with the word month, for example monthyy. The separator characters dash (-), slash(/), or period (.) can be used in the date; however, they must appear in the actual date format. For example, the format for 12-3-07 would be m-d-y.

DETACH

You can use the string 'month' in place of 'm', in which case the month will be name of the month rather than the number. For example, the format for Dec/3/07 would be Month/d/y.

FILLER

The field is ignored.

NUMERIC

The field is a numeric field.

PACKED [UPPERCASE] [NOSPACES]

The field is in IBM/370 or VAX packed decimal format.

TEXT [UPPERCASE] [NOSPACES]

The field is a text field.

[UPPERCASE] [NOSPACES]

If you know that the external data is all uppercase and has no internal spaces in any dimension fields, then use the UPPERCASE and NOSPACES keywords to save CPU. If you do not specify these keywords, they are not used by default.

<width>

Width of the field in fixed and free format records. For free format records, the width determines the amount of buffer space that Application Server will allocate for the incoming data in the Work database.

Note: If you specify a width that is larger than the true field width, the work database might run out of space. If you specify a width that is smaller than the true field width, Application Server truncates the value.

<decimals>

Number of decimal places in a numeric field, in the format .<n> or (<n>), for example, 10.2 or 10(2). The default number of decimal places is two.

With WRITE, the form .<n> overrides the default number of decimal places. For example, if you specify 10.3, numbers will be 10 digits with 3 decimal places. If you specify 10.0, numbers will be 10 digits with no decimal places.

With READ, use the form (<n>) to specify implied decimal places. For example, the digits 12345 read with the specification 5(2) are read as 123.45. If you do not want any decimal places, specify 5 or 5.0.

If decimal points are physically present in the data being read, you do not need to specify <decimals>.

<offset>

Starting position for a field, in a fixed format record. The first position of a record is 1. If you do not specify an offset value, the default value is the position immediately following the previous field.

Note: When more than a few fields are being defined, you should enclose DESCRIPTION in a BEGIN-END block in a procedure and execute the procedure from the Access subsystem.

Example

... Product is a 16-character text field beginning in column 12 of a fixed format record.

... It is followed by the numeric field Sales, which is 10 characters wide with 2 decimal places:

BEGIN

DESCRIPTION

Product TEXT 16 12

Sales NUMERIC 10.2

END

6.54 DETACH

Use

DETACH detaches a database, preventing users from accessing sets and data in it.

Note: You should detach a database before you increase the database partitions using the CHANGE DATABASE command.

Syntax

DETACH [<database>][.]

Parameter	Description
<database>	Name of the database to detach.
.	A period (.) is a synonym for &USEDATABASE.

Example 1: DETACH

...This example detaches the database MARKET and then shows all attached databases:

DETACH market

SHOW DATABASE

Example 2: DETACH

...This example detaches a database to increase the number of database partitions. The database is ... then used and extended so that the database size equals the increased partition sizes.

DETACH Accnt

CHANGE DATABASE Accnt ADD 2 PARTITIONS SIZES 40K, 20K

...The output of this command would be:

Warning: The Number of Blocks Specified in the PARTITION

SIZES (120000) for Database ACCNT is Larger than the

BLOCKS specified.

Database: ACCNT

Maxaccess: UPDATE Disposition: ENABLED

Current State: Available Protection Key: None

Last user: Last access:

Blksize: 8192 Observations 500

Maxblocks: 60000 Number Free: 59990

Database ACCNT Partition Information.

Number of Partitions: 5

Blocksize: 8192

Date Last Opened for Update: Thu Feb 9 15:34:10 2000

Date Last Closed After Update: Thu Feb 9 15:34:10 2000

Partition	Size	Name
1	20000	ACCNT01
2	20000	ACCNT02
3	20000	ACCNT03
4	20000	ACCNT04
5	20000	ACCNT05

USE Accnt

EXTEND Accnt 60000

DIFF()

6.55 DIFF()

Use

DIFF returns the difference between the value of a variable in the current period and in the same period a specified time period ago, (for example, a year or month ago), or in a different time period, using the <period> parameter.

Syntax

CALCULATE <result> = DIFF(<variable>[, <period>])

6.56 Using DIFF and ACCUM

The ACCUM function accumulates data over a period of time. The default is yearly. The following statement accumulates the sales values over the fiscal year. When the next year starts, the function starts over. You get year-to-date figures by default.

ACCSALES=ACCUM(SALES)

If you enter the following command, Application Server returns the original sales, because it handles the change at the start of the fiscal year:

DIFFSALES=DIFF(ACCSALES)

The optional second argument overrides the yearly default, forcing some other periodicity, such as quarterly. However, if you specify ACCUM(<variable>,MONTHLY) for a monthly variable, Application Server accumulates the data over a span of one month, which in effect does not accumulate any data.

Similarly, if you enter DIFF(<variable>, MONTHLY), you are asking Application Server to reset the difference process every month. Application Server returns the original series you gave the function as an argument.

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable whose difference you want to measure. Variable names that use special characters should be in single quotation marks (' ').
<period>	Overrides the default period (year) to compare by quarter, month, or week.

Example 1: DIFF

...Calculate the difference between this period's sales and sales in the same period last year:

CALCULATE Change = DIFF(Sales)

Example 2: DIFF

...Calculate the change in Sales for the current period and the same period in the previous quarter:

CALCULATE Q_Change = DIFF(Sales, Quarter)

Example 3: DIFF

...Accumulate the value of x over a year and uses the DIFF command to reset the difference as z:

USE ron

CREATE x

...The output of this command would be:

1 Variable Created

SET PERIOD 1999

CALCULATE x = obs()

CALCULATE y= accum(x)

CALCULATE z=diff(y)

SELECT VARIABLE

...The output of this command would be:

3 Variables Currently Selected

across variable down time

Across List: # Selected

VARIABLES 3 X, Y, Z

Down List:

TIME

...The output of this command would be:

Period 1999/1/1 - 1999/12/31

Attached Databases: RON User: ADMIN

LIST

...The output of this command would be:

	X	Y	Z
Jan 99	1.00	1.00	1.00
Feb 99	2.00	3.00	2.00
Mar 99	3.00	6.00	3.00
Apr 99	4.00	10.00	4.00
May 99	5.00	15.00	5.00
Jun 99	6.00	21.00	6.00
Jul 99	7.00	28.00	7.00
Aug 99	8.00	36.00	8.00
Sep 99	9.00	45.00	9.00
Oct 99	10.00	55.00	10.00
Nov 99	11.00	66.00	11.00
Dec 99	12.00	78.00	12.00

1 Page of Data Listed

6.57 DIMENSION

Use

DIMENSION starts the Application Server editor, where you can create or edit dimensions. A dimension consists of input members, output members, and a result member that identify the time series for variables in a database. The dimension contains arithmetic rules for these members that define relationships and consolidation methods.

Note: If you manually create a dimension *and* the first OUTPUT member in the consolidation rules must be enclosed in single quotation marks (' ') because it contains special characters, the last OUTPUT or RESULT member must have a label. If you omit the label, Application Server assumes that the member in quotation marks in the rules section is the label for the last OUTPUT or RESULT member.

DIMENSION

DIMENSION - Rules

A dimension does not have to have output members or a result member.

You must define a dimension before creating a variable that is dimensioned by it.

You can use all logic set functions/constructs in output and result member calculations.

When a variable's dimension is omitted from the across/down list, the dimension's result member is used in reports. For example, assume the selected variable SALESVOL is dimensioned by Product. If you enter ACROSS Time DOWN Variables followed by DISPLAY, Application Server displays only the result member of the dimension Product for the variable SALESVOL.

If you create labels for dimension members, the first 24 characters of each label must be unique.

Syntax

DIMENSION [<dimension>] [;<database>| ;EXTERNAL|LOCAL]

Parameter	Description
<dimension>	Name of a new or existing dimension up to 96 bytes. If you do not specify a name, Application Server uses the default dimension (if defined) or the last dimension you edited.
<database>	Name of the database where the dimension is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the dimension is a text file that is not in an Application Server database.
LOCAL	Indicates that the dimension is a text file on the client. Note: You can create or edit a dimension with the EXTERNAL or LOCAL extension, but you must copy it into a model and compile it before it can be used in the model.
	See also these Dimension editor commands:
ALLOCATE	Specifies the maximum number of input, output, and result members in a dimension. If you later add members to the dimension, Application Server will not reorganize the database.
KEY BOTH	Optimizes the selection of dimension members by indexing the members. This occurs when you select by specific member name or label.
INPUT	Specifies the input members, which are the lowest level of data collected or entered in a database.
OUTPUT	Specifies the output members, which are the intermediate consolidated (or calculated) members.
RESULT	Specifies the result member, which is the final consolidated member. There can be only one result member.
LEVEL HIERARCHY	LEVEL defines the members and levels within a dimension. You can select a group of members by specifying the level name instead of selecting the members individually. When a dimension has LEVEL statements without a HIERARCHY statement, members have only one path for rolling up into output levels. Such a dimension has only one hierarchy. HIERARCHY defines the members and levels within a hierarchy in a dimension that contains multiple hierarchies. You define hierarchies when the input members are consolidated into more than one output member.
CLASS	Defines a class or group of dimension members within a dimension. This allows you to select dimension members by their class name instead of selecting them individually.
Consolidation statement	Arithmetical statements that define how Application Server consolidates input and output members and defines their relationships between one another. For example: total_region = SUM, Northeast, Southeast, Midwest, West

Note: If a dimension uses subtraction, you must use the sum operator with a negative sign in the dimension set. For example:

VARIANCE = SUM ACTUAL, -BUDGET

Example

...This example creates a dimension Country, with labels for the dimension members:

DIMENSION Country

INPUT US 'United States'

,UK 'United Kingdom'

RESULT Total 'Total Market'

...The text labels United States, United Kingdom, and Total Market appear on lists and reports

... instead of the member names US, UK, and Total (unless you enter SET SHORT).

...Note: If you create labels for dimension members, the first 24 characters of each label must be unique.

6.58 DIRECTORY

Use

DIRECTORY displays the sets in a database.

Syntax

```
DIRECTORY {DATABASE <database>} [SORT {SIZE | UPDATED} [REVERSE]]
      {FULL}
      {<settype> [...<settype>]}
      {<setname>}
      {SYNC}
      FORMAT string, string
```

Parameter	Description
DATABASE	Displays all sets in the specified database.
<database>	Name of the database.
FULL	Displays information about the size of each set in the database, when Application Server last updated it, and whether you can recover it. If you have made changes to a set during a session, the letter R appears under the Recover category, indicating that you can restore the set to the state it was in when you started the session.
<settype>	Displays all sets of the specified set types in the database. You can specify more than one set type. For example, DIRECTORY REPORT LOGIC ATTRIBUTE lists all reports, logic sets, and attribute sets.
SORT	Sorts and lists sets. You can use SORT options with any DIRECTORY keyword. For example, DIRECTORY FULL SORT SIZE or DIRECTORY LOGIC SORT UPDATE REVERSE.
SIZE	Sorts sets from largest to smallest.
UPDATED	Sorts sets from most recently updated to least recently updated.
REVERSE	Reverses the sort order.
<setname>	Displays all the sets with the specified name in the database. You can use an asterisk (*) in a set name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name. For example, prod* searches for all sets whose name begins with prod.
SYNC	Compares database elements (dimensions, report, logics, and so on) for compile synchronization.

DISABLE (Supervisor)

FORMAT Prints out the specified string for each set found, substituting the set name in the string wherever it finds a '%s'.

string String text.

Example

...This example displays information about all sets:

DIRECTORY FULL

...The output of this command would be:

Recover Procedures	Size	Last Updated
Logics	PROFILE	2 Fri Apr 24 11:38:41 2000
	PROFIT	2 Sun Apr 12 16:13:30 2000
	REVENUE R	2 Fri Apr 24 14:38:28 2000
	SALES	2 Fri Apr 17 14:35:55 2000
Reports	YEARLY	2 Sat Apr 18 09:11:16 2000
Dimensions	COMPANY	2 Sun Apr 19 14:25:59 2000
	PRODUCT	2 Fri Apr 17 16:22:02 2000

...This example uses the asterisk (*) to display all sets that begin with the letters *prof*:

DIRECTORY prof*

...This example displays all the sets in the attached database LIBDB:

DIRECTORY DATABASE libdb

6.59 DISABLE (Supervisor)

Use

DISABLE is a Supervisor command that prevents users from logging in to Application Server or suspends access to a database. Application Server does not remove these records from MASTERDB.

Syntax

DISABLE {LOGINS <users> | DATABASE <databases>}

Parameter	Description
LOGINS <users>	Prevents the specified user(s) from logging in to Application Server, where <users> is one or more user names separated by commas. You can use an asterisk (*) in a user name to indicate that any character can occupy that position or any of the remaining positions in the name.
DATABASE <databases>	Suspends access to the specified database(s). One or more database names separated by commas. You can use an asterisk (*) in a database name to indicate that any character can occupy that position or any of the remaining positions in the name.

Example

...This example disables two users, Fin1 and Fin2:

DISABLE LOGINS Fin1, Fin2

6.60 DISPLAY

Use

DISPLAY displays a report.

Syntax

```
DISPLAY [<report>] [WIDTH <width>] [LENGTH <length>]
        [<periodicity>] [PERIOD <daterange>] [TAB '<symbol>']
        [ RESOLVE ]
```

Parameter	Description
<report>	Name of an existing report. If you do not specify a name, Application Server displays the default report (if defined) or the last report you edited.
WIDTH <width>	Sets the maximum number of characters that can appear on a line.
LENGTH <length>	Sets the maximum number of lines that can appear on a page.
<periodicity>	Displays data at one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, or hourly.
PERIOD <daterange>	Limits the display to a specified date range, where <daterange> is a range of dates separated by a dash (-). For example, 00/12/3 - 00/12/31 is the period from December 3, 2000 through December 31, 2000, and 99 - 00 is all of 1999 and 2000.
TAB '<symbol>'	Uses the character specified by <symbol> as a column delimiter. You must enclose the symbol in single (' ') or double (" ") quotation marks. Use TAB '<symbol>' when the report set uses the TABS keyword but you want to use a different symbol so that you can easily transfer the report into another application. Note: If you use commas as the thousands separator, enter SET DEFAULT COMMA OFF so that Application Server does not display the thousands character. This prevents the thousands character from being interpreted as a column delimiter.
RESOLVE	If you specify this parameter, Application Server recompiles the report before execution. This parameter ensures that Application Server uses the current value of control variables in the set.

Examples: DISPLAY

...This example displays the report **EXPREP** with a monthly periodicity:

```
DISPLAY exprep monthly
```

...This example displays the report **ABCREP** with a weekly periodicity for

...the period from June 1, 1999, to September 30, 1999:

```
DISPLAY abcrep weekly PERIOD 99/6 - 99/8/30
```

...This example displays the report **EXCELREP** with commas for column

...delimiters so that you can transfer the report into Microsoft Excel:

```
DISPLAY excelrep TAB ','
```

6.61 DISPLAY (Logic)

Use

DISPLAY shows the value of a Scalar variable in a logic set.

DOCUMENT

Syntax

DISPLAY <variables>

Parameter	Description
<variables>	One or more Scalar or Temp variable names separated by commas.

DISPLAY (Logic) - Rules

You cannot abbreviate DISPLAY in a logic set.

DISPLAY cannot be the first command in a logic set.

Example

...This example displays the value of the scalar variable *i* in a logic set:

SCALAR i

DO i = 1, 3

 DISPLAY i

ENDDO

...The output from this logic set looks like this:

I 1

I 2

I 3

6.62 DOCUMENT

Use

DOCUMENT starts the Document editor, where you can create or edit documents. You can use DOCUMENT to examine sets created by OUTPUT.

Syntax

DOCUMENT [<document>] [;<database> | ;EXTERNAL|LOCAL]

Parameter	Description
<document>	Name of a new or existing document. If you do not specify a name, Application Server uses the default document. If you do not have a default report, Application Server uses the last document edited.
<database>	Name of the database where the document is located. If you do not specify a database, DOCUMENT uses the database specified in the last USE command.
EXTERNAL	Indicates the document is a server text file that is not in an Application Server database. If the document is a DOS file, its name cannot have an extension.
LOCAL	Indicates that the document is a client text file.

Examples: DOCUMENT

...This example opens an external file "AUTOEXEC.BAT":

DOCUMENT 'C:\AUTOEXEC.BAT';EXTERNAL

...This example creates and opens a document TEXTFILE in the current use database:

DOCUMENT TEXTFILE

6.63 DO-ENDDO

Use

DO-ENDDO repeatedly executes a block of statements for a specified series of values.

Syntax

```
DO <variable> = <m>, <n>
    .
    . <statements>
    .
ENDDO
```

Parameter	Description
DO	Indicates the start of a loop. Executes the statements in the loop as long as the value of the specified variable is within the range of <m> to <n>.
<variable>	Name of the variable assigned the values <m>, <m> + 1, ..., <n>; the value is incremented by 1.
<statements>	A sequence of statements or commands.
ENDDO	Indicates the end of a loop.

Example

...This example executes the statements in the DO-ENDDO block 10 times:

```
DO x=1, 10
    ...your statements
ENDDO
```

6.64 DUMP

Use

Transfers the Use database to one or more external text files (dump files). You can use these dump files with the LOAD command to recreate the database. **Note:** Application Server can dump files greater than 2Gb on UNIX).

Syntax

```
DUMP <filename> [DATA <datafile>] [SIZES <sizes>] [DUMMY] [NODATA] [OVERWRITE] [DRILLTHRU]
[NODENSE] [NOLARGEFILES] [CHARSET {ASCII | ISO2022 | SHIFTJIS | JEUC | TCEUC | SCEUC |
KEUC | UTF8 }]
```

Parameter	Description
<filename>	Specifies the alphanumeric name of the external file in which to store the entire database, including data, dimensions, procedures, and so on. By default, one dump file will be created. The file can exceed 2Gb. If you don't want the dump file to exceed 2Gb, you can partition it by specifying a name containing wildcard characters and using the NOLARGEFILES and/or SIZES keywords. You can choose to include just the database definition within <filename>, and place data within another file specified by the <datafile> parameter. If you choose to do this, you cannot specify wildcards within <filename>.
DATA <datafile>	Stores just the data from the database in an external file with the name specified by <datafile>. The database definition is stored in <filename>. By default, one dump file will be created. The file can exceed 2Gb. If you don't want the dump files to exceed 2Gb, specify a file name containing wildcards and use the NOLARGEFILES and/or SIZES keywords.

DUMP

SIZES <sizes>	<p>Specifies the size of the file, or files, represented by the <datafile> parameter. To control the sizes of multiple partitions, specify a list of sizes separated by commas. If you specify fewer sizes than there are partitions, the last value is used as the size for the remaining partitions. You can specify the sizes in any of the following units:</p> <p>Note: When specifying the SIZES parameter, the values cannot be greater than 2Gb if expressed as raw (unscaled numbers). To specify a SIZE greater than 2Gb you must use the K, M, or G scaling factors.</p> <p>Bytes, for example 3000000000 Kilobytes, for example 1500000K Megabytes, for example 4000M Gigabytes, for example 5G</p> <p>If you have omitted the <datafile> parameter, <sizes> applies to the <filename> parameter.</p>
NODENSE	<p>If you want to dump a database and make sure it is backward compatible so that it can be loaded in a version of Application Server prior to version 9.4, you should use the NODENSE keyword. This prevents the dump file containing explicit commands to create DENSE measures, as this keyword would be rejected in any versions prior to 9.4.</p>
DUMMY	<p>Replaces all nonmissing data values with 1. This option is useful if you do not want to dump your actual data.</p>
NODATA	<p>Dumps the definition of the variables, but not the data values.</p>
OVERWRITE	<p>Overwrites a system file with the same name.</p>
DRILLTHRU	<p>In addition to the time series stored in Application Server, writes data from all Hybrid OLAP schema tables into the dump file. When you subsequently load the data from the dump file, all data, regardless of its origin, is loaded into Application Server as non-drillthru data.</p>
NOLARGEFILES	<p>Creates the dump files without exceeding 2Gb. If the dump file is larger than 2Gb, then partitions will be created.</p> <p>Use the NOLARGEFILES keyword with the SIZES keyword to restrict the file sizes to a specify size less than 2Gb. Do not use NOLARGEFILES with a SIZES value that is greater than 2Gb.</p> <p>You might want to restrict file sizes to 2Gb for any of these reasons:</p> <ul style="list-style-type: none"> • You might want to move the files to another machine or operating system that does not support large files. • The dump size is too big to fit on a single disk (or the free space on a single disk) and must be split over multiple disks.
CHARSET	<p>Identifies that the output dump file should be encoded by a particular charset. Application Server converts the dump text from an internal charset to the parameter charset. If you omit the CHARSET keyword, Application Server uses EXTOUT charset to do the conversion. The DUMP command adds a SET CHARSET EXTIN charset line at the beginning of the dump file and SET CHARSET EXTIN SESSION at the end of the dump file.</p>

Remarks

DUMP copies binary object sets to EXTERNAL files with a name consisting of the first letter of the binary type (for example, 'B' for BINARY, 'M' for MULTIMEDIA) followed by the set name, and inserts a COPY command into the dump file to copy them back to the database. The DUMP command does not compress binary object sets.

Creating a dump file with multiple partitions

By default, one large dump file is created. It can exceed 2Gb up to 64-bit. If you do not want a database dump to exceed 2GB, you can split the dump across multiple files (partitions). To dump a database to multiple partitions, you specify a wildcard within the dump file name, for example DUMP??. You can also specify NOLARGEFILES to make sure that each partition is less than 2Gb. You can also specify the SIZES keyword to specify the exact sizes of each partition. Application

Server replaces the wildcard characters with partition numbers, for example DUMP01, DUMP02, and so on. If you have chosen to store the data and database definition together, you should specify the wildcard within the <filename> parameter; if you have chosen to store the data and database definition separately, you should specify the wildcard within the <datafile> parameter. For example:

DUMP dump?? SIZE 1G

DUMP dump?? NOLARGEFILES

DUMP maindump DATA dump?? SIZE 1G

Location of dump files in Windows

In Windows, dump files are placed in the Home directory. However, you can specify alternative locations by editing the LSSERVER.INI file before executing the DUMP command:

```
[Windows]
dump##=pathname\dump##
```

Substitute dump## with the name of a dump file partition, and substitute pathname with the directory into which you want the dump file to be placed. You should create a separate entry for each partition, for example:

```
[Windows]
dump01=c:\db\dump01
dump02=d:\db\dump02
dump03=e:\db\dump03
```

Location of dump files in UNIX

In UNIX client/server, dump files are placed in the directory defined by the LSSHOME environment variable. To specify a different location for the dump files, add environment variables to lssstcp.sh, for example:

```
dump##=pathname/dump##
export dump##
```

Substitute dump## with the name of a dump file partition, and substitute pathname with the directory into which you want the dump file to be placed. You should create a separate entry for each partition, for example:

```
dump01=/db1/dump01
dump02=/db2/dump02
dump03=/db2/dump03
export dump01
export dump02
export dump03
```

Rules

If you load a file into a database that is not empty, the sets or variables loaded overwrite any existing sets or variables with the same name.

File and datafile are alphanumeric names, but the first character must be alphabetic.

Due to its size, you may not be able to edit the dump file. In this case, either separate your data from the rest of the database, or use SNAPSHOT to dump a subset of your database.

Example

...This example dumps the contents of the database CORPFIN to an

...external file FINANCE and then loads that file into the database NEWFIN:

USE corpfm

DUMP (Supervisor)

DUMP finance

USE newfin 'the new, empty database

LOAD finance

6.65 DUMP (Supervisor)

Use

DUMP is a Supervisor command that creates an external file on your system containing all the Supervisor commands needed to recreate MASTERDB. Application Server encrypts user passwords on the dump file. You can restore MASTERDB using the SUPERVISOR LOAD command.

Note: If you are trying to load a dump of MASTERDB that was dumped prior to version 9.3 of Application Server and contains information about partitioned PAS databases, the dump file will be unloadable. You must edit the dump file to remove the PARTITION and NAME clauses from the ADD DATABASE commands, and then you will be able to load the pre-9.3 version of MASTERDB. For help doing this contact Customer Support.

Syntax

DUMP <filename> [CREATE|DEFINE] [OVERWRITE]

Parameter	Description
<filename>	Name of the external file where you want to store the contents of MASTERDB.
CREATE	Adds CREATE DATABASE commands to the dump file so that when you load the dump file, Application Server creates the database.
DEFINE	Adds DEFINE DATABASE commands to the dump file. Application Server creates the databases when you enter the first explicit or implicit USE command.
OVERWRITE	Overwrites a system file with the same name.

Rules

DUMP creates an external file with the Supervisor commands necessary to recreate MASTERDB in its current state. The file has commands to recreate database records and user records.

For database records, the file contains ADD DATABASE commands to add the record for that database and any partitions in MASTERDB. The file containing the database must already exist on the system. Following ADD DATABASES is a comment with the number of observations, the number of blocks, and the block size of the database.

For user records, the first command removes the user and the next command recreates the user with the characteristics of that user as those that exist in the dumped MASTERDB; Application Server replaces any existing records with the same names as those in the dump file.

You must be in the Supervisor subsystem to restore the external file created with DUMP.

Application Server dumps access keys, and user and database entries, but does not dump protection keys.

Example

...This example dumps the contents of MASTERDB to an external file named MSTRBAK:

DUMP mstrbak

...In the event that the file containing MASTERDB is destroyed, you can restore

...MASTERDB with the following SUPERVISOR command:

LOAD mstrbak

6.66 ECHO

Use

ECHO displays a message on either the OUTPUT, REASSURANCE, or ERROR streams. The default stream is to display the message on the OUTPUT stream.

Syntax

ECHO [ERROR | REASSURANCE] <message>

Parameter	Description
ERROR	Specifies that Application Server will display the message on the ERROR stream.
REASSURANCE	Specifies that Application Server will display the message on the REASSURANCE stream.
<message>	Message text of up to 255 characters.

Note: To include a percent sign (%) in the output generated by the ECHO command, use two percent symbols. For example, the following command:

ECHO A%%B

generates the output:

A%B

Example

...This example displays the message "Executing Stage 3" on the default OUTPUT stream:

ECHO Executing Stage 3

...This example displays the message "Stage 3 failed!" on the ERROR stream:

ECHO ERROR Stage 3 failed!

6.67 ENABLE (Supervisor)

Use

ENABLE is a Supervisor command that restores a user's access either to Application Server or to a database.

Syntax

ENABLE {LOGINS <users> | DATABASE <databases>}

Parameter	Description
LOGINS <users>	Restores the specified user's access to Application Server, where <users> is one or more user names separated by commas. You can use an asterisk (*) in a user name to indicate that any character can occupy that position or any of the remaining positions in the name.
DATABASE <databases>	Restores access to the specified database(s), where <databases> is one or more database names separated by commas. You can use an asterisk (*) in a database name to indicate that any character can occupy that position or any of the remaining positions in the name.

Example

...This example enables two users, Fin1 and Fin2:

ENABLE LOGINS Fin1, Fin2

END

6.68 END

Use

END exits and saves all changes made in the current session to the Access, Rollup, and Supervisor subsystems.

Syntax

END [LEAVE]

Parameter	Description
LEAVE	Ends the Access subsystem and keeps the connection to the RDBMS open so that the next time you go back to the Access subsystem, it is still connected. If you are going in and out of ACCESS LSLINK over and over, you can avoid connecting and disconnecting by using the LEAVE statement in ACCESS LSLINK.

6.69 EXECUTE

Use

EXECUTE executes the commands in a procedure, or in an external file or tape device, without displaying them on your screen.

Syntax

EXECUTE <procedure> <setname>[:<database>] | <filename>[:EXTERNAL] | TAPE '<tapename>' [UPDATE] [RESOLVE]

Parameter	Description
<setname>	Name of a set, such as a logic set or a dimension set.
<procedure>	Name of a procedure. If you do not specify a name, Application Server uses the default procedure (if defined) or the last procedure you edited.
<filename>	Name of an external file that Application Server created by the SUPERVISOR DUMP command that contains user and database information about MASTERDB.
<database>	Name of the database where the procedure is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the procedure is a text file that is not in an Application Server database.
TAPE	Indicates that the data source is a tape device.
"<tapename>"	The UNIX name for the tape drive in double quotation marks (" ").
UPDATE	Forces a database update after each command in the procedure. By default, EXECUTE freezes the database until Application Server completes the procedure, and then updates it. EXECUTE runs faster if you do not specify UPDATE. However, if you do not specify UPDATE, and the procedure causes an error, none of the commands in the procedure update the database.
RESOLVE	If you specify this parameter, Application Server recompiles the procedure before execution. This option ensures that Application Server uses the current value of control variables in the set.

EXECUTE - Rules

If you do not want to include any of the EXECUTE options, you can execute a procedure by typing its name and then clicking the OK button.

Examples: EXECUTE

...This example executes the procedure ENDYEAR:

EXECUTE endyear

...Optionally, you can execute the procedure without specifying EXECUTE. For example:

endyear

...This example executes the DOS/Windows file c:\myjob:

EXECUTE c:\myjob;external

...This example executes the procedure FRED on the USE database, if found,

...otherwise Application Server searches for a file called FRED.

EXECUTE FRED

...This example executes the external file FRED in the current directory:

EXECUTE ./FRED

6.70 EXHIBIT

Use

EXHIBIT displays information about attributes, databases, dimension members, sets, users, User-Defined Hierarchies, variables and virtual variables. EXHIBIT provides a mechanism for interrogating the contents of a database. You can use the EXHIBIT TOPS command to exhibit and select the top members of all dimensions.

Output Formats

The output for requests that return object information is formatted in a list format with the information followed by a line feed (ASCII 10). For example, EXHIBIT <database> returns:

APLIB

DEMO1

INITIAL

JUICE

SMREPORT

The output for requests that return property-related information is formatted as a tab-separated (ASCII 9) string. For example, EXHIBIT GROUP TEST FULL returns:

TEST ADMIN TEST1 TEST2 SUPER SUPERVISOR

If you request property-related information for multiple objects, a tab-separated string terminating in a line feed is returned for each object. For example, EXHIBIT GROUP FULL returns:

TEST ADMIN USER1 USER2 SUPER SUPERVISOR

TEST2 USER1 USER2

6.70.1 EXHIBIT ACROSS

Displays the across dimensions.

Note that attribute names are displayed with a leading underscore (_), as shown in the example.

Example

... This example displays the current across dimensions in the JUICE database:

USE JUICE

EXHIBIT ACROSS

... The output of this command would be:

EXHIBIT

TIME

PRODUCT

_FLAVOR

... Note that FLAVOR is an attribute, and is displayed with a leading underscore.

6.70.2 EXHIBIT DOWN

Displays the down dimensions.

Note that attribute names are displayed with a leading underscore (_), as shown in the example.

Example

... This example displays the current down dimensions in the JUICE database:

USE JUICE

EXHIBIT DOWN

... The output of this command would be:

VARIABLES

PRODUCT

_FLAVOR

... Note that because FLAVOR is an attribute, the name is displayed with a leading underscore (_).

6.70.3 EXHIBIT ADIMENSION [attribute] [EXCLUDE] [BASIS | FULL]

Lists all attribute dimensions.

Parameter	Description
<attribute>	A specific attribute for which you want to display details. Omit this parameter to display details for all attributes.
EXCLUDE	Checks for the existence of a document set called EXCLUDEDIM in the Use database; the document specifies a list of dimensions and attributes that should be excluded from the EXHIBIT list. If you do not specify EXCLUDE, details of all attributes are listed. Note: If a dimension is listed in EXCLUDEDIM, its attributes will still be shown by EXHIBIT ADIMENSION. To exclude attributes, as well as dimensions, from the EXHIBIT list, you must specify them in EXCLUDEDIM.
BASIS	Displays the name of the attribute, followed by the base dimension.
FULL	Displays full details for the attribute or attributes in tab-separated format. The fields are as follows: Attribute name Base dimension Maximum number of input members Actual number of inputs Maximum number of outputs Actual number of outputs Maximum number of result members (zero or one) Actual number of result members (zero or one) Number of levels

Comma-separated list showing the number of members in each level (ordered from input to result)

Number of selected members

Maximum number of User-Defined Hierarchies

Actual number of User-Defined Hierarchies

Example

... This example displays the name of each attribute in the JUICE

... database, followed by the base dimension name:

USE JUICE

EXHIBIT ADIMENSION BASIS

... The output of this command would be:

CONTAINER	PRODUCT
COT	CUSTOMER
FLAVOR	PRODUCT
PARENT	CUSTOMER
SIZE	PRODUCT
STATE	CUSTOMER

6.70.4 EXHIBIT [ALL] [SELECTED] MEMBERS...

Syntax

EXHIBIT [ALL] [SELECTED] MEMBERS <dimension> [HIERARCHY <hierarchy>] [<keyword>] [BOTH] [POSITION] [ALLHIERARCHIES]

Where <keyword> can be one of the following:

```
[
  TOPS
  LEVELS [<level>]
  <pattern>
  INPUTS
  OUTPUTS
  RESULT
  ABOVE <member>
  BELOW <member>
  DRILL ABOVE <member>
  DRILL BELOW <member>
  JUST ABOVE <member>
  JUST BELOW <member>
]
```

Parameter	Description
ALL	Modifies the command to display all the members of a dimension, regardless of any set hierarchy. Note: The ALL keyword must directly follow the EXHIBIT command, as in EXHIBIT ALL.
SELECTED	Lists the selected members of the specified dimension. If the dimension contains multiple hierarchies, it lists the selected members of the currently set hierarchy within the dimension. However, if used with the ALL keyword, it displays all members of all hierarchies.

EXHIBIT

	If used with the HIERARCHY statement, it lists the selected members of the specified hierarchy, regardless of whether it is the currently set hierarchy.
MEMBERS <dimension>	Lists the members of the specified dimension. If the dimension contains multiple hierarchies, it lists all the members of the currently set hierarchy of the specified dimension (unless used with the ALL keyword).
HIERARCHY <hierarchy>	Lists the members of the specified hierarchy, regardless of whether it is the currently set hierarchy in the dimension.
BOTH	Displays both the dimension member names and labels. The BOTH keyword ignores the SET SHORT LONG setting.
POSITION	Displays the numeric order of the dimension members. This is not valid for the Variables dimension.
ALLHIERARCHIES	Forces the EXHIBIT DIMENSION command to work on all the dimension's hierarchies. By default, the EXHIBIT DIMENSION command always works on the current hierarchy or the hierarchy specified by the HIERARCHY keyword.
TOPS	Lists the dimension's top member, which normally is the result. If the dimension has no result member, Application Server displays the first member from the highest output level. If the dimension has no outputs and no result, the first input member is displayed. If your dimensions are declared with ALLOCATE or CUSTOM statements that reserve space for additional members, you can use the keyword to display the top member of an individual dimension (EXHIBIT DIMENSION dimname TOP). You cannot use this keyword to get a list of the top members for all dimensions (EXHIBIT TOP and EXHIBIT TOP SELECT setname).
LEVEL [<level>]	Lists the dimension's levels. If you specify a level, it lists the members in that level. If the dimension contains multiple hierarchies, LEVEL lists all the hierarchy level names in the dimension. If you specify a level, it lists the members of the specified hierarchy (unless used with the ALL keyword).
<pattern>	Lists the dimension's members based on the specified wildcard pattern.
INPUTS	Lists all the dimension's input members. If the dimension contains multiple hierarchies, INPUT lists only the input members that are part of the currently set hierarchy (unless used with the ALL keyword).
OUTPUTS	Lists all the dimension's output members. If the dimension contains multiple hierarchies, OUTPUT lists only the output members that are part of the currently set hierarchy (unless used with the ALL keyword).
RESULT	Lists the result member of the specified dimension.
ABOVE <member>	Lists the specified member and the members of all levels above it. If the dimension contains multiple hierarchies, member is a member of the currently set hierarchy. ABOVE member lists that member name and all the members above it in the same hierarchy (unless used with the ALL keyword). To exhibit above a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.
BELOW <member>	Lists the specified member and all members at levels below it. If the dimension contains multiple hierarchies, member is a member of the currently set hierarchy. BELOW member lists that member name and all the members below it in the same hierarchy (unless used with the ALL keyword). To exhibit below a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.
DRILL ABOVE <member>	Lists all members in the level above this member, but not including this member. If the dimension contains multiple hierarchies, DRILL ABOVE member lists all the members in the currently set hierarchy above this member, but not including the member (unless used with the ALL keyword). To drill above a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.
DRILL BELOW <member>	Lists all members in the level below this member, but not including this member.

If the dimension contains multiple hierarchies, DRILL BELOW member lists all the members in the same hierarchy below it, but not including that member (unless used with the ALL keyword).

To drill below a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.

JUST ABOVE <member> Lists the specified member and all members one level above it.

If the dimension contains multiple hierarchies, JUST ABOVE member lists the member from the currently set hierarchy and the members in the same hierarchy one level above it (unless used with the ALL keyword).

To exhibit just above a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.

JUST BELOW <member> Lists the specified member and all members one level below it.

If the dimension contains multiple hierarchies, JUST BELOW member lists the member from the currently set hierarchy and the members in the same hierarchy one level below it (unless used with the ALL keyword).

To exhibit just below a member that is not in the currently set hierarchy, specify the alternate hierarchy name using a HIERARCHY statement.

6.70.5 EXHIBIT ATTRIBUTES [dimension]

This command displays the names of dimensions that have attributes, or the names of attributes for a specified dimension.

Parameter	Description
<dimension>	The name of the dimension for which you want to display attributes. Omit this parameter to display the names of dimensions that have attributes.

Example

... This example displays the attributes of the Product dimension in the JUICE database:

USE JUICE

EXHIBIT ATTRIBUTES PRODUCT

... The output of this command would be:

CONTAINER

FLAVOR

SIZE

6.70.6 EXHIBIT BASIS attribute [BOTH]

Displays the structural dimension for the specified attribute dimension.

Parameter	Description
<attribute>	Specifies the name of the attribute dimension.
BOTH	Displays both the structural dimension and attribute dimension names.

Example

... This example displays both the structural dimension and attribute

... dimension names for the Flavor attribute in the JUICE database:

USE JUICE

EXHIBIT BASIS FLAVOR BOTH

... The output of this command would be:

EXHIBIT

PRODUCT FLAVOR

... If you omit the BOTH keyword, only the structural dimension name is displayed:

EXHIBIT BASIS FLAVOR

... The output of this command would be:

PRODUCT

6.70.7 EXHIBIT CHARSET

Use

EXHIBIT CHARSET displays the current CHARSET in effect from a SET CHARSET command.

Syntax

EXHIBIT CHARSET [INPUT | INTERNAL | OUTPUT | EXTIN | EXTOUT | SESSION]

Result

Outputs will be one of the following: ASCII, ISO2022, SHIFTJIS, JEUC, TCEUC, SCEUC, KEUC or UTF8

Example

EXHIBIT CHARSET INPUT

Display the current INPUT setting.

EXHIBIT CHARSET INTERNAL

Display the current INTERNAL setting.

EXHIBIT CHARSET OUTPUT

Display the current OUTPUT setting.

EXHIBIT CHARSET EXTIN

Display the current EXTIN setting.

EXHIBIT CHARSET EXTOUT

Display the current EXTOUT setting.

EXHIBIT CHARSET SESSION

Display the current PAS process' SESSION CHARSET.

EXHIBIT CHARSET

Display all CHARSET setting. One output example:

INPUT: UTF8

INTERNAL: SCEUC

OUTPUT: UTF8

EXTIN: SCEUC

EXTOUT: SCEUC

SESSION: UTF8

6.70.8 EXHIBIT CHECKPOINT

Returns the current checkpoint status as UPDATE, FREEZE, or FREEZE CONTINUE.

Example

... This example displays the current checkpoint status:

USE JUICE

EXHIBIT CHECKPOINT

... The output of this command would be:

UPDATE

6.70.9 EXHIBIT COMPILED DIMENSIONS

Lists all the dimensions that have been compiled.

Note that attribute names are displayed with a leading underscore (_) character, as shown in the example.

Example

... This example displays a list of the compiled dimensions in the JUICE

... database, with attribute dimensions prefixed by an underscore (_):

USE JUICE

EXHIBIT COMPILED DIMENSIONS

... The output of this command would be:

CHANNEL

CUSTOMER

PRODUCT

_CONTAINER

_COT

_FLAVOR

_PARENT

_SIZE

_STATE

6.70.10 EXHIBIT COMPTIME FROM database

Displays the compiled time sets in a specified database.

Parameter	Description
<database>	Specifies the name of the database.

Example

... This example displays the timesets in the DEMO1 database:

USE DEMO1

EXHIBIT COMPTIME FROM DEMO1

...as follows:

WKLY1

WKTREND

WY

EXHIBIT

WYL

6.70.11 EXHIBIT CONTROL variable

Displays the value of a specified control variable.

Parameter	Description
<variable>	Specifies the name of the control variable.

Example

... This example displays the value of the control variable Selection:

```
USE JUICE
```

```
EXHIBIT CONTROL selection
```

6.70.12 EXHIBIT COUNT [dimension | attribute] [BYLEVEL [SECURITY [FULL]]] [DRILLTHRU]

Use

Displays the model's counts as specified by the keywords.

Syntax

```
EXHIBIT COUNT [<dimension> | <attribute>] [BYLEVEL [SECURITY [FULL]]] [DRILLTHRU]
```

Parameter	Description
COUNT	Lists the counts for allocated inputs, allocated outputs, allocated result, total selected members, defined User-Defined Hierarchies, and available User-Defined Hierarchies. Note: If you execute a basic EXHIBIT COUNT command with no keywords, the output represents all ALLOCATED numbers, independent of any security.
<dimension> <attribute>	Specifies the name of a dimension or attribute.
BYLEVEL	Displays the number of members at each level of the dimension. If used with the SECURITY keyword, it only shows counts for the members at each level that have security applied.
SECURITY	Shows counts for the members at each level that have security applied by a Security procedure. If a level does not have secured members, that column representation does not appear in the output.
FULL	Shows counts for the members at each level that have security applied by a Security procedure. If a level does not have secured members, that column is represented with a 0.
DRILLTHRU	Hybrid OLAP only. Reads the dimension counts from the RDBMS to ensure that the full counts are displayed and not just the number of members that have been imported into Application Server.

Example

... In this example, for the PRODUCT dimension in the JUICE database, the counts for allocated inputs, allocated outputs, allocated result, total selected members, defined User-Defined Hierarchies and available User-Defined Hierarchies are returned:

```
USE JUICE
```

```
EXHIBIT COUNT PRODUCT
```

... The output of this command would be:

```
36  17  1      1      5  5
```

... There are 36 allocated input members, 17 allocated output level members, one allocated result member, two selected members, and five defined and available User-Defined Hierarchies.

... If you use the BYLEVEL keyword, as in this example, the return value lists the number of levels in the dimension, followed by a count of the members in each level:

EXHIBIT COUNT PRODUCT BYLEVEL

... The output of this command would be:

```
4    36  9      3      1
```

... The PRODUCT dimension has four levels - the top level, and the levels PRODUCT, BRAND, and

... CATEGORY. The PRODUCT (input) level has 36 members, the BRAND level has nine members, the

... CATEGORY level has three members, and the top level has one member.

6.70.13 EXHIBIT CUSTOM dimension [SELECTED] [PRESENT]

Use

Displays the User-Defined Hierarchies that belong to a specified dimension and identifies whether any dimensions in the across/down list have any User-Defined Hierarchies selected.

Syntax

EXHIBIT CUSTOM { PRESENT | <dimension> [SELECTED | FORMAT '<%s text>'] }

Parameter	Description
PRESENT	Displays a 0 or 1 that identifies whether any dimensions in the current across/down list have any User-Defined Hierarchies selected. A value of 0 means that no dimensions in the across/down have any User-Defined Hierarchies selected. A value of 1 means that at least one dimension in the across/down has a User-Defined Hierarchy selected.
<dimension>	Specifies the name of a dimension whose User-Defined Hierarchies you want to view.
SELECTED	Displays just the selected User-Defined Hierarchies for this dimension.
FORMAT '<%s text>'	Display the specified text appended to the User-Defined Hierarchy names in the %s location. This allows you to quickly build a set of similar commands. For example: EXHIBIT CUSTOM Product FORMAT 'SELECT %s' SELECT UDH1 SELECT UDH2

Example

... This example displays a list of the currently defined User-Defined Hierarchies for the PRODUCT dimension in the JUICE database:

EXHIBIT CUSTOM PRODUCT

... The output of this command would be:

```
NEW_PRODUCTS
```

6.70.14 EXHIBIT DATABASES EXCDOCDB

Like the EXHIBIT DATABASES command, displays the properties of a specified database, or all databases, provided that the database(s) is not listed in an exclusion document held in SMREPORT. The exclusion document contains a list of databases that the user is not permitted to access.

The name of the exclusion document is based on the user name, and takes the form EXCLUDEDDB<username>. If no exclusion document is found for the current user, a default exclusion document, EXCLUDEDDBDEFAULT, is used instead.

Note: The output of this command is the same as for the EXHIBIT DATABASES command.

6.70.15 EXHIBIT DATABASES [database] [FULL] [HIDDEN]

Displays the properties of a specified database, or all databases.

Parameter	Description
<database>	Specifies the name of a database. Omit this parameter to display all databases.
FULL	<p>When specified with the <database> option, displays the properties of that database. Otherwise, displays the properties of all databases registered in MASTERDB. The <database> option returns the properties in tab-separated format in the following order:</p> <p>Dimensional Model</p> <p>File Name</p> <p>No. Partitions</p> <p>Last Access</p> <p>Status</p> <p>State</p> <p>Update User</p> <p>Default Usage</p> <p>Max Access</p> <p>Max Blocks</p> <p>Block Size</p> <p>Max Size</p> <p>Blocks Free</p> <p>% Blocks Free</p> <p>Max Variables</p> <p>Max Members</p> <p>Max Observations</p> <p>Multiples</p> <p>Input Multiples Only</p> <p>Notes:</p> <ul style="list-style-type: none"> Some of these fields are described in the CREATE DATABASE topic. Some fields are informational, and cannot be set directly. "Last Access" is displayed in the format: DDDD/MM/YY HH:MM:SS
HIDDEN	Includes details of databases created using the CREATE DATABASE <database> NOEXHIBIT command, which are not normally displayed by the EXHIBIT DATABASES command.

Examples

... This example displays a list of all databases in MASTERDB:

EXHIBIT DATABASES

... The output of this command would be:

APLIB

DEMO1

INITIAL

JUICE

SMREPORT

... This example displays the properties of the specified database, JUICE:

USE JUICE

EXHIBIT DATABASES JUICE FULL

... The output of this command would be:

```
JUICE  C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\JUICE.ENG      1      2000/04/02 14:51:07
        Enabled      In use (read)ADMIN  Read-only  Update      5000 8192 32  4521
        90.42        0      10000  1000 0      No
```

... This example displays the properties of all the databases in MASTERDB:

EXHIBIT DATABASES FULL

... The output of this command would be:

```
APLIB  C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\APLIB      1      2000/04/02 12:39:37      Enabled
        Available      ADMIN      Read-only  Update      10000 512 32  4885 48.85 0
        10000          5000      0      No

DEMO1  C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\DEMO1      1      2000/03/26 13:13:09      Enabled
        Available      ADMIN      Read-only  Update      10000 8192 32  9041 90.41 0
        10000          1000      0      No

INITIAL C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\INITIAL      1      2000/03/19 16:45:04      Enabled
        Available      ADMIN      Read-only  Update      10000 4096 32  9968 99.68 0
        10000          5000      0      No

JUICE  C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\JUICE.ENG      1      2000/04/02 14:51:07
        Enabled      In use (read)ADMIN  Read-only  Update      5000 8192 32  4521
        90.42        0      10000  1000 0      No

SMREPORT C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY MANAGEMENT\
        APPLICATIONSERVER\DATA\SMREPORT      1      2000/02/25 11:21:36      Enabled      Available
        ADMIN      Shared      Update 10000 2048 32  9969 99.69 0  10000 5000 0
        No

TBDB   C:\PROGRA~1\SAP BUSINESSOBJECTS\STRATEGY
        MANAGEMENT\APPLICATIONSERVER\DATA\TBDB.ENG      1      Enabled      Available
        Exclusive  Read-only  600 8192 0  90 15.00 0  0  0
        0      No
```

6.70.16 EXHIBIT DATABASES [USE | WORK | TEMPORARY | UPDATE] [FULL]

Displays information about databases.

Parameter	Description
DATABASES	Displays the names of all databases registered in MASTERDB.
USE	Displays the name of the Use database.
WORK	Displays the name of the Work database.
TEMPORARY	Displays the name of any temporary database.
UPDATE	Displays all databases in MASTERDB that have MAXACCESS set for update, and are not offline or disabled.
FULL	Displays the full path location of the database and the database name.

Example

... This example displays a list of all the databases currently listed in MASTERDB:

EXHIBIT DATABASES

... The output of this command would be:

APLIB

DEMO1

INITIAL

JUICE

SMREPORT

... Specify the USE keyword to display the name of the current Use database. For example:

EXHIBIT DATABASES USE

... The output of this command would be:

JUICE

... Specify the WORK keyword to display the name of the current Work database. For example:

EXHIBIT DATABASES WORK

... The output of this command would be:

WKADMIN

... Specify the UPDATE keyword to list databases in MASTERDB that have MAXACCESS set for

... update, and are not offline or disabled. For example:

EXHIBIT DATABASES UPDATE

... The output of this command would be:

APLIB

DEMO1

INITIAL

JUICE

SMREPORT

6.70.17 EXHIBIT DATABASES database [READKEYS | UPDATEKEYS | SORTKEYS]

Note: This command is available only to SUPERVISOR users.

Displays access keys for a specified database.

Parameter	Description
<database>	Specifies the name of a database.
READKEYS	Displays the read keys (if any exist) associated with the database. The Supervisor must add a read key to a user's record to enable that user to read from the database.
UPDATEKEYS	Displays the update keys (if any exist) associated with the database. The Supervisor must add an update key to a user's record to enable that user to update the database.
SORTKEYS	Displays the dimension order method used for the internal Application Server database key structures. A value of "Alphabetic" or "Nmembers" is displayed based on whether dimensions are ordered alphabetically or based on the number of members in each dimension. You specify the dimension order when issuing the CREATE DATABASE command. Note: Knowing the SORTKEYS method is useful when dumping and loading databases. If you load a dump file from an NMEMBERS model into an ALPHABETIC database or vice versa, the rollup quadrants will not be the same for each model. Procedures with rollup commands that reference specific quadrants may need to be updated. In addition, users need to make sure that all databases that make up a virtual cube model have the same key order.

Note: Access keys are added with the SUPERVISOR ADD ACCESS command, and removed with the SUPERVISOR REMOVE ACCESS command.

Examples

... **Note:** You must be logged on as SUPERVISOR in order to use this command. This example

... returns a list of read keys for the DEMO1 database:

EXHIBIT DATABASES DEMO1 READKEYS

... The output of this command would be:

RDKEY1

... This example returns a list of update keys for the DEMO1 database:

EXHIBIT DATABASES DEMO1 UPDATEKEYS

... The output of this command would be:

UPKEY1

6.70.18 EXHIBIT DATABASES database USERS [READ | UDPATE | CONNECTED]

Displays the users of a database.

EXHIBIT

Parameter	Description
<database>	Specifies the name of a database.
USERS	With the READ option, displays users who can attach to the database for read access. With the UPDATE option, displays users who can attach to the database for update access.
READ	Displays users with permission to read from the specified <database>. This is the default value.
UPDATE	Displays users with permission to update the specified <database>.
CONNECTED	Displays a list of the users who are currently connected to the specified <database>.

Note: You can also view details of user privileges in Application Server.

Examples

... This example displays a list of users for the JUICE database:

USE JUICE

EXHIBIT DATABASES JUICE USERS

... The output of this command would be:

ADMIN

INITIAL

SUPER

SUPERVISOR

TEST1

... This example lists users with read access to the JUICE database:

USE JUICE

EXHIBIT DATABASES JUICE USERS READ

... The output of this command would be:

ADMIN

INITIAL

SUPER

SUPERVISOR

TEST1

... This example lists users with update access to the JUICE database:

USE JUICE

EXHIBIT DATABASES JUICE USERS UPDATE

... The output of this command would be:

ADMIN

INITIAL

SUPER

SUPERVISOR

... USERS CONNECTED displays a list of the users currently connected to the database.

EXHIBIT DATABASES JUICE USERS CONNECTED

... The output of this command would be:

ADMIN

TEST1

6.70.19 EXHIBIT DATABASES database ACCESS

Display the current access mode assigned to a dimensional model.

Parameter	Description
<database>	Specifies the name of a database.
For example, if the Juice database has an access mode of Shared, and you type: EXHIBIT DATABASE JUICE ACCESS the system will return: SHARED	

6.70.20 EXHIBIT DAYS

Displays the names of days, using the currently selected language.

Example

... This example returns the day names in English:

EXHIBIT DAYS

... The output of this command would be:

Sunday

Monday

Tuesday

Wednesday

Thursday

Friday

Saturday

Example

... This example returns the name of the member, followed by three tab-separated numbers that

... denote the number of parents, number of children, and the level of the member:

EXHIBIT DIMENSION PRODUCT DETAIL

... The output of this command would be:

Seltzer Water 0 3 3

...This example returns the number of parents, children, and the level of the member

EXHIBIT DIMENSION PRODUCT BELOW b1c1 DETAIL

Courtyard 12oz Conc. 1 0 1

Courtyard 12oz 1 0 1

Courtyard 16oz 1 0 1

Courtyard 36oz 1 0 1

Courtyard 48oz 1 0 1

EXHIBIT

Courtyard 64oz 1 0 1
 Courtyard 1 6 2

...This example returns a 1 to represent that the members have parents

EXHIBIT DIMENSION PRODUCT BELOW c1 POSITION BOTH PARENTS

1 UPC1B1C1 Courtyard 12oz Conc. 1
 2 UPC2B1C1 Courtyard 12oz 1
 3 UPC3B1C1 Courtyard 16oz 1
 4 UPC4B1C1 Courtyard 36oz 1
 5 UPC5B1C1 Courtyard 48oz 1
 6 UPC6B1C1 Courtyard 64oz 1

...This example returns the number 0 to show that the members do not have children

EXHIBIT DIMENSION PRODUCT BELOW c1 POSITION BOTH CHILDREN

1 UPC1B1C1 Courtyard 12oz Conc. 0
 2 UPC2B1C1 Courtyard 12oz 0
 3 UPC3B1C1 Courtyard 16oz 0
 4 UPC4B1C1 Courtyard 36oz 0
 5 UPC5B1C1 Courtyard 48oz 0
 6 UPC6B1C1 Courtyard 64oz 0

6.70.21 EXHIBIT DIMENSION dimension

**[keyword] [POSITION] [BOTH]
 [SORTED [REVERSE [LONG]]]
 [MDXNAME] [SECURITY] [RANGE
 <number>-<number>] [DETAIL
 [CUSTOM]] [ALLHIERARCHIES]
 [LEVEL [FULL]]**

Use

Displays information about dimensions.

Syntax

EXHIBIT DIMENSION <dimension> [<keyword>] [POSITION] [BOTH] [SORTED [REVERSE
 [LONG]]] [MDXNAME] [SECURITY] [RANGE <number>-<number>] [DETAIL [CUSTOM]]
 [ALLHIERARCHIES] [LEVEL [FULL]]

Where <keyword> can be one of the following:

[SELECTED
 UNSELECTED
 INPUTS
 OUTPUTS
 RESULT
 TOPS
 LEVELS [<level>]
 CLASSES [<class>]
 DRILLTHRU
 ABOVE <member> [COUNT]

BELOW <member> [COUNT]
 JUST ABOVE <member> [COUNT]
 JUST BELOW <member> [COUNT]
 ONLY BELOW <member> [COUNT]
 ONLY JUST BELOW <member> [COUNT]
 ONLY ABOVE <member> [COUNT]
 ONLY JUST ABOVE <member> [COUNT]
 HIERARCHY [<hierarchy>]
 FORMAT '<text> %s'
 CHILDREN
 ORPHANS
 PARENTS
 <pattern> [MATCH {SHORT|LONG}]]

Parameter	Description
<dimension>	Displays members of the specified dimension. If SET SHORT is in effect, Application Server displays the dimension's member names. If SET LONG is in effect, Application Server displays the dimension's member labels.
POSITION	Displays the numeric order of the dimension members. This option is not valid for the variables dimension. Note: The default selection for a dimension is ALL, meaning that Application Server selects all members.
BOTH	Displays both short and long names for the specified dimension. The BOTH keyword ignores the SET SHORT/LONG setting, as appropriate.
SORTED	Sorts all the members of the dimension in alphabetical order from a to z by the short names. Note: You can sort by long names by including the BOTH and LONG keywords.
REVERSE	Sorts all the members of the dimension in alphabetical order from z to a by the short names.
LONG	Sorts the long names of the selected dimension members. When using the SORTED LONG keywords, you must also use the BOTH keyword, which displays both short and long names. For example, EXHIBIT DIMENSION product BOTH SORTED LONG.
SECURITY	Filters the exhibited list by displaying only those items that have security applied in a Security procedure. Note: It is not necessary to use the SECURITY keyword on the SELECTED, UNSELECTED, INPUTS, OUTPUTS, RESULTS, TOPS, pattern, ABOVE, and BELOW, and variants of those keywords because the output from those keywords are already displayed based on any security applied.
RANGE <number>-<number>	Displays the dimension members within the range of numbers. For example, EXHIBIT DIMENSION product INPUTS RANGE 1-10 displays the first ten input members from the Product dimension.
DETAIL	Displays the number of parents, the number of children, and the level of each member. If you use DETAIL with the SECURITY keyword, the output will only show counts of children and parents that you are actually allowed to see according to the Security procedure. The children are only the descendants that are visible through the security definitions, recursing as deep down the hierarchy as necessary. For example if there are children directly below total_customer but those children are excluded by security, then they will not be displayed in the EXHIBIT output. The following Security procedure: <pre>type proc security index user</pre>

EXHIBIT

```

CASE guest
... User can see all stores and overall totals
SEL CUSTOMER INPUT
SEL CUSTOMER PLUS 'TOTAL CUSTOMER'
SEL SALES,COSTS
endindex

```

Would yield results like this:

```

SELECT Customer RESULT
1 Member of CUSTOMER Selected
EXHIBIT DIMENSION Customer SELECTED DETAIL
TOTAL CUSTOMER 0 3 4
EXHIBIT DIMENSION Customer SELECTED DETAIL SECURITY
TOTAL CUSTOMER 0 83 4
SELECT Customer ONLY JUST BELOW Total_Customer
0 Members of CUSTOMER Selected
SEL Customer ONLY JUST BELOW Total_Customer SECURITY
83 Members of CUSTOMER Selected

```

CUSTOM	Displays additional detail about whether the member is a User-Defined Hierarchy. The identifier is the last value in the detail output. A value of 0 means that it is not a User-Defined Hierarchy and a value of 1 means that the member is a User-Defined Hierarchy.
ALLHIERARCHIES	Forces the EXHIBIT DIMENSION command to work on all the dimension's hierarchies. By default, the EXHIBIT DIMENSION command always works on the current hierarchy.
LEVEL	Lists the dimension's user-defined level names. If a dimension has a level without a user-defined name, that level will not be listed unless you use the FULL keyword. If the dimension contains multiple hierarchies, it lists the user-created levels in the currently set hierarchy, in the order in which you define the levels.
FULL	Lists all the dimension's level names, including user-defined names and names generated by Application Server for the unnamed levels. Application Server-generated level names have the form \$<num>\$<num> where the first number is the hierarchy number and the second number is the level number.
SELECTED	Lists the selected members of the specified dimension.
UNSELECTED	Lists the unselected members of the specified dimension.
INPUTS	Lists all the dimension's input members. If the dimension contains multiple hierarchies, INPUT lists only the input members that are part of the currently set hierarchy.
OUTPUTS	Lists all the dimension's output members. If the dimension contains multiple hierarchies, OUTPUT lists only the output members that are part of the currently set hierarchy.
RESULT	Lists the result member of the specified dimension. If a dimension contains multiple hierarchies, you cannot use RESULT because a dimension with multiple hierarchies does not have a result member.
TOPS	Lists the dimension's top member, which normally is the result. If the dimension has no result member, Application Server displays the first member from the highest output level. If the dimension has no outputs and no result, the first input member is displayed. If your dimensions are declared with ALLOCATE or CUSTOM statements that reserve space for additional members, you can use the keyword to display the top member of an individual dimension (EXHIBIT DIMENSION <dimname> TOP). You cannot use this keyword to get a list of the top members for all dimensions (EXHIBIT TOP and EXHIBIT TOP SELECT <setname>).
LEVELS [<level>]	Lists the dimension's levels. If the dimension contains multiple hierarchies, LEVELS returns the members of the hierarchy's level. It shows only the members of a level in the currently set hierarchy. Specify <level> if you want to display the members at that level.

	<p>If you use LEVEL with the SECURITY keyword, the output will only show levels that have secured members. It is not necessary to use LEVEL [level] with SECURITY because the displayed members have already been filtered by security.</p>
CLASSES [<class>]	<p>Lists the dimension's classes. Specify <class> if you want to display all the members in that class.</p> <p>If you use CLASS with the SECURITY keyword, the output will only show classes that have secured members. It is not necessary to use CLASS [class] with SECURITY because the displayed members have already been filtered by security.</p>
DRILLTHRU	<p>For Hybrid OLAP dimensions only. Displays the dimension members stored in Application Server, and also displays dimension members stored in Hybrid OLAP schema tables.</p> <p>Note: If you do not specify the DRILLTHRU keyword for a Hybrid OLAP dimension, just the members that are stored in Application Server are displayed. If you use the NO_GUESTS and MAX_GUEST_LEVEL keywords in the Dimension table and you use an EXHIBIT ABOVE BELOW DETAIL PARENTS CHILDREN keyword that accesses a member stored in the RDBMS, then the EXHIBIT command is forced to query the RDBMS to complete the command.</p>
ABOVE <member>	<p>Lists the specified member and all members in the same hierarchy above it.</p> <p>If you use the SECURITY keyword with ABOVE or BELOW or any of the variants of those keywords, you will get the descendants or ascendants that are accessible with security applied. Application Server will recurse through as many levels of the dimension as necessary to skip descendants/ascendants that are excluded by security to do this. For example:</p> <pre> type proc security index user CASE guest ... User can see all stores as well as overall totals. SEL CUSTOMER INPUT SEL CUSTOMER PLUS 'TOTAL CUSTOMER' SEL SALES,COSTS endindex EXH DIM Customer ONLY JUST BELOW Total_Customer EXH DIM Customer ONLY JUST BELOW Total_Customer SECURITY RANGE 1-10 7 Eleven - Nashua, NH Costco - Hudson, NH CVS - Lowell, MA CVS - Providence, RI Market Basket - Chelmsford, MA RITE-AID - Worcester, MA Shaw's - Framingham, MA Stop & Shop - Andover, MA Stop & Shop - Newton, MA Stop & Shop Burlington, MA </pre>
COUNT	<p>Counts the number of members resulting from the EXHIBIT command. For example, EXHIBIT DIMENSION customer BELOW region COUNT.</p>
BELOW <member>	<p>Lists the specified member and all members in the same hierarchy below it.</p>
JUST ABOVE <member>	<p>Lists the specified member and the members of the same hierarchy one level above.</p>
JUST BELOW <member>	<p>Lists the specified member and the members of the same hierarchy one level below.</p>

EXHIBIT

MDXNAME	When displaying data from an SAP NetWeaver BI InfoCube, this keyword exhibits the MDX MEMBER_UNIQUE_NAME immediately after the Application Server short and/or long name in the output.
ONLY BELOW <member>	Lists all members in the same hierarchy below this member, but not including the specified member.
ONLY JUST BELOW <member>	Lists only the members in the path directly below the specified member.
ONLY ABOVE <member>	Lists all members in the same hierarchy above this member, but not including the specified member.
ONLY JUST ABOVE <member>	Lists only the member in the path directly above the specified member.
HIERARCHY <hierarchy>	Lists the members of the specified hierarchy, regardless of whether it is the currently set hierarchy in the dimension.
FORMAT '<text> %s %d'	Display the specified text with the dimensions appended in the %s location. Use %d to represent decimal output. This allows you to quickly build a set of similar commands. For example, FORMAT 'SELECT %s' displays dimensions with the SELECT command in front of them. You can also use the syntax FORMAT '%s text'. You must supply the same number of placeholders that will appear in the EXHIBIT output. For example, EXHIBIT DETAIL outputs a name, the number of parents, the number of children and the level. You must supply four %s characters.
CHILDREN	Displays either a 0 or 1, indicating whether this member has children. 0 represents that the member does not have children. 1 represents that the member has children. If you use CHILDREN with the SECURITY keyword, the output will show a 0 or 1 depending on whether any children or parents exist and have security applied.
ORPHANS	Shows the input and output members that do not have a parent (do not belong to a hierarchy) within the specified dimension.
PARENTS	Displays either a 0 or 1, indicating whether this member has parents. 0 represents that the member does not have a parent. 1 represents that the member has a parent.
<pattern>	Lists the dimension's members based on the specified wildcard pattern. The wildcard finds dimension member matches using the short names of the dimension member whenever the short name is present in the output (ie, whenever you use the BOTH keyword or use the SET SHORT command or use the SET SELECT NAME command). You can override this by using the MATCH LONG keyword. The wildcard finds dimension member matches using the long names if you are using the SET LONG command or using the SET SELECT LABEL command <i>and</i> you are not using the BOTH keyword on the EXHIBIT command. You can override this by using the MATCH SHORT keyword.
MATCH SHORT	Allows wildcard matching to occur on the short names of dimension members. This will override any SET LONG or SET SELECT LABEL commands.
MATCH LONG	Allows wildcard matching to occur on the long names of dimension members. This will override any SET SHORT or SET SELECT NAME commands.

Examples

... This example displays all members of the FLAVOR dimension in the JUICE database:

USE JUICE

EXHIBIT DIMENSION FLAVOR

... The output of this command would be:

APPLE

BLUEBERRY

CHERRY
 CRANBERRY/APPLE
 CRANBERRY/GRAPE
 GRAPE
 GRAPE/APPLE
 LEMON
 LEMON/LIME
 ORANGE
 RASPBERRY
 RASPBERRY/GRAPE
 STRAWBERRY
 TOTAL FLAVOR

6.70.22 EXHIBIT DIMENSION [EXCLUDE] [FULL] [ALLHIERARCHIES]

Lists dimension details.

Parameter	Description
DIMENSION	Displays names of all dimensions (both structural and attribute) in your Use database.
EXCLUDE	<p>Checks for the existence of a document set called EXCLUDEDIM in the Use database; the document specifies a list of dimensions that should be excluded from the EXHIBIT list. If you do not specify EXCLUDE, details of all dimensions and attributes are listed.</p> <p>Note: If a dimension is listed in EXCLUDEDIM, its attributes will still be shown by EXHIBIT DIMENSION. To exclude attributes, as well as dimensions, from the EXHIBIT list, you must specify them in EXCLUDEDIM.</p>
FULL	<p>Displays full details for the dimensions in tab-separated format. The fields are as follows:</p> <ul style="list-style-type: none"> Dimension name Maximum number of input members Actual number of inputs Maximum number of outputs Actual number of outputs Maximum number of result members (0 or 1) Actual number of result members (0 or 1) Number of levels Comma-separated list showing the number of members in each level (ordered from input to result) Number of selected members Maximum number of User-Defined Hierarchies Actual number of User-Defined Hierarchies
ALLHIERARCHIES	Forces the EXHIBIT DIMENSION command to work on all the dimension's hierarchies. By default, the EXHIBIT DIMENSION command always works on the current hierarchy.

EXHIBIT

Example

... This example displays details of all dimensions in the JUICE database:

USE JUICE

EXHIBIT DIMENSION FULL

... The output of this command would be:

CHANNEL	3	2	2	0	1	1	2	2,1	3	0	0
CONTAINER	4	4	6	0	1	1	2	4,1	5	5	0
COT 4	4	6	0	1	1	2	4,1	5	5	0	
CUSTOMER	84	83	18	12	1	1	4	83,9,3,1	96	5	0
FLAVOR	13	13	6	0	1	1	2	13,1	14	5	0
PARENT	24	24	6	0	1	1	2	24,1	25	5	0
PRODUCT	37	36	18	12	1	1	4	36,9,3,1	49	5	0
SIZE 6	6	6	0	1	1	2	6,1	7	5	0	
STATE 10	10	6	0	1	1	2	10,1	11	5	0	

6.70.23 EXHIBIT FISCAL

Displays a description of your Use database's fiscal year. Indicates if the database is lunar.

Example

... This example displays a description of the fiscal year of your Use database, in this example,

... JUICE. The first line indicates the fiscal pattern, the second line indicates the first month in the

... fiscal year, the third line indicates the first day in the fiscal year, and the fourth line indicates if the

... database is lunar (the line is blank if the database is not lunar).

USE JUICE

EXHIBIT FISCAL

... The output of this command would be:

CALENDAR

JANUARY

SUNDAY

6.70.24 EXHIBIT GROUP

[<user_defined_hierarchy>] [FULL]

Displays information about user groups.

Parameter	Description
<user_defined_hierarchy>	Displays the name of a User-Defined Hierarchy if it already exists. Omit this parameter to display all User-Defined Hierarchies.
FULL	Displays the members of the User-Defined Hierarchies. Returns a tab-separated string in the following format: Group Name \tUser 1 \tUser 2 \t...

Example

... This example lists the existing User-Defined Hierarchies in the TEST database:

USE TEST

EXHIBIT GROUP

... The output of this command would be:

MANAGE

TEST

... This example lists the members of the MANAGE User-Defined Hierarchies in the TEST database:

USE TEST

EXHIBIT GROUP MANAGE FULL

... The output of this command would be:

MANAGE ADMIN ROBBINST01 SMITHW01 SUPERVISOR

6.70.25 EXHIBIT HIERARCHY

Lists the dimensions in the database, followed by the number of hierarchies in each dimension.

The first output column represents the number of hierarchies; the second column identifies the dimension name.

Example

... This example shows the number of hierarchies in all dimensions. STORES has three hierarchies, and both of the other dimensions have only one hierarchy:

EXHIBIT HIERARCHY

... The output of this command would be:

3 STORES

1 PRODUCT

1 TYPE

6.70.26 EXHIBIT HIERARCHY dimension

Lists the names of the hierarchies in a specified dimension, in the order in which you defined the hierarchies.

Parameter	Description
<dimension>	Specifies the dimension name.

Example

... This example displays the names of the hierarchies for the PRODUCT dimension in the DEMOMH database, and shows the order in which the hierarchies were added:

USE DEMOMH

EXHIBIT HIERARCHY PRODUCT

... The output of this command would be:

1 PRODUCTLINE

2 MANUFACTURER

... In the JUICE database, the PRODUCT dimension has a single hierarchy:

USE JUICE

EXHIBIT HIERARCHY PRODUCT

EXHIBIT

... The output of this command would be:

1 DEFAULT

6.70.27 EXHIBIT SELECTED HIERARCHY dimension

Lists the currently set hierarchy name and number for the specified dimension.

Note: The default hierarchy is the one defined in the first CONSTRUCT HIERARCHY statement. The currently set hierarchy is either the default hierarchy or a hierarchy that overrides the default using the SET command.

Parameter	Description
<dimension>	Specifies the name of the dimension.

Example

... In this example, **PRODUCTLINE** is the currently set hierarchy defined for the **PRODUCT** dimension in the **DEMOMH** database. **PRODUCTLINE** is the first hierarchy in the **PRODUCT** dimension.

USE DEMOMH

EXHIBIT SELECTED HIERARCHY Product

... The output of this command would be:

1 PRODUCTLINE

6.70.28 EXHIBIT INDEXES

Displays the names of variables in the Use database, and their internal index numbers.

Example

... This example lists the indexes and their internal index

... numbers in the Use database, in this case, **JUICE**:

USE JUICE

EXHIBIT INDEXES

... The output of this command would be:

```

11 Container
14 Costs
8 Class of Trade
12 Flavor
10015 Margin
9 Parent
16 Quota
17 Sales
13 Size
10 State
18 Units
10019 Variance
```

6.70.29 EXHIBIT LEVELS <dimension> [HIERARCHY <hierarchy>] [FULL]

Parameter	Description
LEVELS <dimension>	<p>Lists the dimension's user-defined level names.</p> <p>If a dimension has a level without a user-defined name, that level will not be listed unless you use the FULL keyword.</p> <p>If the dimension contains multiple hierarchies, it lists the user-created levels in the currently set hierarchy, in the order in which you define the levels.</p>
HIERARCHY <hierarchy>	<p>Lists the level names within the specified hierarchy of the dimension, where <hierarchy> can be any hierarchy name or hierarchy order number in the dimension, regardless of the currently set hierarchy. The hierarchy's level names are listed in the order in which they were defined.</p>
FULL	<p>Lists all the dimension's level names, including user-defined names and names generated by Application Server for the unnamed levels. Application Server-generated level names have the form \$<num>\$<num> where the first number is the hierarchy number and the second number is the level number.</p>

Examples

The output of this command:

USE DEMO1

EXHIBIT LEVELS PRODUCT

would be:

- 1 DEVICE
- 2 CATEGORY

The output of this command:

USE DEMO1

EXHIBIT LEVELS PRODUCT FULL

Would be:

- 1 DEVICE
- 2 CATEGORY
- 3 \$1\$4

This example shows the levels of the second hierarchy in the PRODUCT dimension in the DEMOMH database. VENDORCLASS is the lowest level:

... The output of this command:

USE DEMOMH

EXHIBIT LEVELS PRODUCT HIERARCHY 2

would be:

- 1 DEVICE
- 2 VENDOR
- 3 VENDORCLASS

6.70.30 EXHIBIT LOGIN

Displays the name of the user currently logged in.

EXHIBIT

Example

... This example displays the name of the current user:

USE JUICE

EXHIBIT LOGIN

... The output of this command would be:

ADMIN

6.70.31 EXHIBIT MEASURE [ONLY] [JUST] BELOW <virtual_variable>

Displays hierarchies of variables and virtual variables in the specified virtual variable.

Parameter	Description
ONLY BELOW	Lists all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also listed. The specified virtual variable is not included in the list.
JUST BELOW	Lists the specified virtual variable and all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is listed, but the variables that make up that virtual variable are <i>not</i> listed.
BELOW	Lists the specified virtual variable and all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also listed.
ONLY JUST BELOW	Lists all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is listed, but the variables that make up that virtual variable are <i>not</i> listed. The specified virtual variable is not included in the list.
<virtual_variable>	Specifies the virtual variable whose variables and virtual variable hierarchies you want to display.

6.70.32 EXHIBIT [MEASURE | VARIABLE | VVARIABLE] <name> <property>

Displays the properties of the measure, variable, or virtual variable.

Parameter	Description
<name>	Specifies the name of the measure, variable, or virtual variable.
<property>	Specifies the property to display for the specified measure, variable, or virtual variable. You can use the following options: AFTER – Displays a suffix (if specified) of up to three characters that is displayed after each value. AUTO – Indicates whether automatic scaling is used to fit the width when displaying large data values. COMMA – Displays the thousands separator and the scale. COMMALocale – Indicates whether the locale setting is used to determine the thousands separator. CURRENCY – Displays the currency symbol, and the scale. DECIMALS – Displays the number of decimal places. DECIMALLocale – Indicates whether the locale setting is used to determine the decimal separator. DIMENSIONS – Displays the dimensions.

EXPENSE – Indicates whether the variable is an expense.

FLOAT – Indicates that the number of decimal places varies as required (rather than always displaying the number of places specified by the DECIMALS setting).

FIXCUR – Indicates whether the currency symbol is displayed at the left edge of the field, or immediately to the left (or right) of the value.

FIXNEG – Indicates whether the negative symbol is displayed at the left edge of the field, or immediately to the left (or right) of the value.

INTEGRAL – Indicates how Application Server stores the data values internally — 1-, 2-, or 4-byte, or 4- or 8-byte floating point variables.

JUXCUR – Indicates whether the currency symbol is displayed after the value (corresponds to the REVERSE setting).

LABEL – Displays the long name.

LINKID – (Hybrid OLAP only) Displays the Link ID that Application Server uses to access data that is stored in a relational database in a Hybrid OLAP configuration.

METHOD – Indicates the time conversion method used to convert the variable from one periodicity to another — sum (default), inclusive average, exclusive average, first value, last value, or weighted on another variable.

MISSING – Displays the scale, and the symbol used for missing values.

NEGATIVE – Displays the scale, and the negative symbol.

NOCONSOLIDATE – Indicates that the variable will not be consolidated by Application Server.

OBSERVATIONS – Displays the variable name and the number of observations.

PAD – Indicates whether values are padded with leading zeros to fill the WIDTH setting.

PERIODICITY – Displays the periodicity.

POINT – Displays the character used as the decimal separator.

POINTLOCALE – Indicates whether the locale setting is used to determine the decimal separator.

PREFIX – (Hybrid OLAP only) Displays the prefix of the Hybrid OLAP schema in which the data is stored.

RANGE – Displays the date range of the variable.

RATE – Indicates that the variable is a rate variable which undergoes time conversion but not consolidation.

SCALE – Displays the number of places by which the variable is scaled (by moving the decimal point to the left). For example, a variable with a scale setting of 1 is converted from 1000.00 to 100.00.

SPARSE – Indicates that the variable contains many consecutive similar values over time (for example, 10, 10, 10, 10, 5, 5, 5). Application Server uses compression to store SPARSE variables more efficiently.

TYPE – Displays the variable type — text, numeric, or virtual.

UNITS – Indicates whether Application Server rounds values to the nearest whole number when performing time conversions.

WIDTH – Displays the maximum number of spaces a value can occupy when displayed (for example, by a LIST command).

ZERO – Displays the character that Application Server uses to display zeros.

Examples

... This example displays the suffix:

EXH VAR TEST AFTER

... returns:

VARIABLE=TEST AFTER=CAL

EXHIBIT

... This example indicates that automatic scaling is used:

EXH VAR TEST AUTO

... returns:

VARIABLE=TEST SCALE=10 AUTO=YES

EXH VAR COST AUTO

... returns:

VARIABLE=COSTS AUTO=YES

... This example displays the thousands separator and the scale.

EXH VAR COST COMMA

... returns:

VARIABLE=COSTS COMMA=,

... This example indicates that COSTS uses the locale setting to determine the thousands separator:

EXH VAR COSTS COMMALocale

... returns:

VARIABLE=COSTS COMMALocale=Yes

... This example displays the currency symbol, and the scale:

EXH VVAR PROFIT CURRENCY

... returns:

VARIABLE=PROFIT CURRENCY=\$ SCALE=10

... This example displays the measure name, the number of decimal places, and the scale:

EXH MEA TEST DECIMALS

... returns:

TEST DECIMALS=2 SCALE=10

... This example indicates that the locale setting is not used to determine the number of decimal places:

EXH VAR COSTS DECIMALLocale

... returns:

VARIABLE=COSTS DECIMALLocale=No

... This example displays the dimensions for COSTS:

EXH VAR COSTS DIMENSIONS

... returns:

VARIABLE=COSTS DIMENSION=CHANNEL, CUSTOMER, PRODUCT

... This example indicates that the TEST variable is an expense:

EXH VAR TEST EXPENSE

... returns:

VARIABLE=TEST EXPENSE=YES

... This example indicates that the currency symbol for the TEST measure is displayed to the left of the field:

EXH MEASURE TEST FIXCUR

... returns:

TEST FIXCUR=YES

... This example indicates that the negative symbol for the TEST measure is displayed to the left of the field:

EXH MEASURE TEST FIXNEG

... returns:

TEST FIXNEG=YES

... This example indicates that COSTS is stored as a 4-byte integral variable:

EXH VAR COSTS INTEGRAL

... returns:

VARIABLE=COSTS BYTES=4

... This example indicates that the currency symbol for COSTS appears after the value:

EXH VAR COSTS JUXCUR

... returns:

VARIABLE=COSTS JUXCUR=YES

... This example displays the long name:

EXH VAR COT LABEL

... returns:

VARIABLE=COT LABEL=Class of Trade

... This example displays the time conversion method used for the TEST measure:

EXH MEASURE TEST METHOD

... returns:

TEST Sum

... This example displays the symbol used for missing values:

EXH VAR COSTS MISSING

... returns:

VARIABLE=COSTS MISSING=-

... This example displays the negative symbol:

EXH VAR COSTS NEGATIVE

... returns:

VARIABLE=COSTS NEGATIVE=-

... This example indicates that COSTS is not consolidated:

EXH VAR COSTS NOCONSOLIDATE

... returns:

VARIABLE=COSTS NOCONSOLIDATE=YES

... This example displays the number of observations:

EXH VAR COSTS OBSERVATIONS

... returns:

VARIABLE=COSTS NOBS=18

... This example indicates that leading zeros are displayed to fill the variable width:

EXH VAR COSTS PAD

... returns:

VARIABLE=COSTS PAD=YES

EXHIBIT

... PERIODICITY

... This example displays the periodicity:

EXH VAR COSTS PERIODICITY

... returns:

VARIABLE=COSTS PERIODICITY=MON

... This example displays the decimal separator:

EXHIBIT VAR COSTS POINT

... returns:

VARIABLE=COSTS POINT=.

... This example indicates that the locale setting is used to determine the decimal separator:

EXH MEA SALES POINTLOCALE

... returns:

SALES POINTLOCALE=Yes

... This example displays the date range:

EXH VAR COSTS RANGE

... returns:

VARIABLE=COSTS RANGE=Jan 96 - Jun 97

... This example displays the scale:

EXH VAR COSTS SCALE

... returns:

VARIABLE=COSTS SCALE=10

... This example indicates that QUOTA uses the SPARSE setting:

EXH MEA QUOTA SPARSE

... returns:

QUOTA SPARSE=YES

... This example indicates that QUOTA is numeric

EXH MEA QUOTA TYPE

... returns:

QUOTA VARTYPE=NUMERIC

... This example indicates that Application Server rounds values to the nearest

... whole number when performing time conversions for TEST:

EXH MEA TEST UNITS

... returns:

TEST UNITS=YES

... This example displays the QUOTA width:

EXH MEA QUOTA WIDTH

... returns:

QUOTA WIDTH=15

... This example displays the scale, and the character used to display zeros:

EXH MEA TEST ZEROS

... returns:

TEST SCALE=10 ZERO=-

6.70.33 EXHIBIT [MEASURE | VARIABLE | VVARIABLE] name BOTH

Displays both the long and short names for the measure, variable, or virtual variable specified by <name>.

Parameter	Description
<name>	Specifies the name of the measure, variable, or virtual variable.
BOTH	Displays both the long and short names for the measure, variable, or virtual variable specified by <name>.

Example

... This example displays the long and short names of the COGS measure:

EXH MEASURE COGS BOTH

... returns:

COGS Cost of Goods

... This example displays the long and short names of the Variance virtual variable:

EXH VVAR VARIANCE BOTH

... returns:

VARIANCE Variance

6.70.34 EXHIBIT MEASURES ...

EXHIBIT MEASURES

```
{<measure> [FULL] }
{ [DATABASE <remotedb>] [FULL] [EXCLUDE] ] [BOTH] [SORT [REVERSE [LONG]]
[MDXNAME] [RANGE <integer> - <integer>]
{ NONVIRTUAL [<pattern>] [DIMENSIONS] }
```

Lists all variables (not including attributes).

Parameter	Description
<measure>	Specifies the name of a measure. Omit this parameter to display all measures. Note: You cannot specify <measure> and DATABASE <remotedb> together.
DATABASE <remotedb>	Displays details of measures contained within distributed remote databases. You can specify a database name or a wildcard character (*) to display details of variables for one or all databases. Note: You cannot specify <measure> and DATABASE <remotedb> together.
EXCLUDE	Lists all variables and virtual variables, and excludes any variables listed in the Excludevar document, if one exists. For example: EXHIBIT MEASURES EXCLUDE – lists all virtual variables and all variables not included in the EXCLUDEVAR document. EXHIBIT MEASURES DATABASE remotedb EXCLUDE –lists all virtual variables and variables in distributed remote databases, and excludes any variables listed in the EXCLUDEVAR document. Notes:

EXHIBIT

- If you also use the FULL option, specify EXCLUDE after FULL.
- You cannot use the EXCLUDE keyword when using EXHIBIT for a single measure.

FULL	<p>Returns the properties of the measure(s) in tab-separated format in the following order:</p> <p>Measure name (short name)</p> <p>Link ID (Hybrid OLAP only)</p> <p>Prefix (Hybrid OLAP only)</p> <p>Periodicity</p> <p>Type (numeric, virtual, text)</p> <p>Label (long name)</p> <p>Rate variable (YES or NO)</p> <p>Sparse (YES or NO)</p> <p>Values are stored as integers (YES or NO)</p> <p>Number of bytes used to store each observation</p> <p>Expense (YES or NO)</p> <p>Not consolidated (YES or NO)</p> <p>Dimensionality</p> <p>Values are rounded during time conversion (YES or NO)</p> <p>Time conversion method (sum, inclusive average, exclusive average, first, last, weighted)</p> <p>Number of observations</p> <p>Date range</p> <p>Width</p> <p>Decimal setting is locale dependent (YES or NO)</p> <p>Number of decimal places</p> <p>Currency symbol</p> <p>Characters displayed after data values</p> <p>Decimal separator is locale dependent (YES or NO)</p> <p>Decimal separator character</p> <p>Thousand separator is locale dependent (YES or NO)</p> <p>Thousand separator character</p> <p>Digit grouping (number of digits between thousands separators)</p> <p>Group setting is locale dependent (YES or NO)</p> <p>Factor by which data values are scaled</p> <p>Negative symbol</p> <p>Values are automatically scaled to fit the width setting (YES or NO)</p> <p>Number of decimal places can reduce below the decimal setting (YES or NO)</p> <p>Currency symbol is displayed in the leftmost position within the field (YES or NO)</p> <p>Negative symbol is displayed in the leftmost position within the field (YES or NO)</p> <p>Negative symbol is displayed before the currency symbol (YES or NO)</p> <p>Values padded with leading zeros to fill the width setting (YES or NO)</p> <p>Character used to represent missing values</p> <p>Formula used to calculate a virtual variable</p> <p>Database name, if used with the DATABASE keyword.</p>
BOTH	<p>Displays both short and long names for the specified measure. The BOTH keyword ignores the SET SHORT/LONG setting, as appropriate.</p>

SORT	Sorts the output from the EXHIBIT command.
REVERSE	Sorts in reverse order (last to first).
LONG	Sorts the long names of the measures. When using the SORT LONG keywords, you must also use the BOTH keyword, which displays both short and long names. For example, EXHIBIT MEASURE BOTH SORT LONG.
MDXNAME	When displaying data from an SAP NetWeaver BI InfoCube, this keyword exhibits the MDX measure name immediately after the Application Server short and/or long name in the output.
RANGE	Limits the sort to the specified range of measures between and including the first integer and the last integer.
NONVIRTUAL	Displays nonvirtual measures.
<pattern>	Displays the virtual or nonvirtual measures with the specific wildcard pattern. Use an asterisk (*) as a wildcard to indicate that any character can occupy that position or any of the remaining positions in the name.
DIMENSIONS	Displays the dimensions associated with the virtual or nonvirtual dimensions.

Example

... This example displays details of the COSTS measure in the JUICE database:

USE JUICE

EXHIBIT MEASURES COSTS FULL

... The output of this command would be:

COSTS	MON	Costs	No	No	No	4	No	Yes				
	CHANNEL, CUSTOMER, PRODUCT					18	Jan 96 - Jun 97	15	No	2		
		Yes	.	Yes	,		-	Yes	No	No		
	No	No	No		-							

... This example creates an Excludevar doc and exhibits all virtual variables and all variables

... not included in the Excludevar:

USE JUICE

DOCUMENT EXCLUDEVAR

Costs

Units

Margin

EXHIBIT MEASURES EXCLUDE

... The output of this command would be:

Quota

Sales

Variance

6.70.35 EXHIBIT MEASURE FROM { <report> | <logic> | <virtual_variable> } [RECURSE] [EXCLUDE] [SORT [REVERSE]]

Parameter	Description
FROM	Displays variables from the specified report or logic set or virtual variable.
<report>	Displays variables from the specified report.

EXHIBIT

<logic>	Displays variables from the specified logic set.
<virtual_variable>	Displays variables from the specified virtual variable.
RECURSE	Displays the variables referenced in the logic, report, or virtual variable. If a virtual variable is used in the specified logic, report, or virtual variable, then RECURSE also displays the variables that compose that virtual variable.
SORT	Sorts the variables in alphabetical order from a to z.
REVERSE	Reverses the sort order of variables so they are listed from z to a.
EXCLUDE	Displays the specified variables, and omits hidden variables.

6.70.36 EXHIBIT {DIMENSION | MEMBER [dimension] SORT ...

Displays information about a dimension.

The full syntax for this command is:

EXHIBIT {DIMENSION [<dimension>] | MEMBER <dimension> } SORT [[REVERSE] RANGE <integer> - <integer>] }

Parameter	Description
DIMENSION	Displays all defined dimensions and attributes in the Use database.
<dimension>	Specifies all the members of this dimension.
MEMBER <dimension>	Displays all the members of the specified dimension.
SORT	Sorts the output from the EXHIBIT command.
REVERSE	Sorts in reverse order (last to first).
RANGE	Limits the sort to the specified range of members between and including the first integer and the last integer.

Examples

... This example displays a list of dimensions in the JUICE database:

USE JUICE

EXHIBIT DIMENSION

... The output of this command would be:

CHANNEL

CONTAINER

COT

CUSTOMER

FLAVOR

PARENT

PRODUCT

SIZE

STATE

... This example lists the members of the PRODUCT dimension in the JUICE database:

USE JUICE

EXHIBIT MEMBER PRODUCT

... The output of this command would be:

Courtyard 12oz Conc.

Courtyard 12oz

Courtyard 16oz
Courtyard 36oz
Courtyard 48oz
Courtyard 64oz
Summer Delight 12oz Grape Conc.
Summer Delight 48oz Grape
Fall Berry 24oz Cran/Apple
Fall Berry 48oz Cran/Apple
Fall Berry 64oz Cran/Grape Conc
Fall Berry 24oz Cran/Grape
Fall Berry 48oz Cran/Grape
Timeless Timber 12oz Apple
Timeless Timber 36oz Apple
Timeless Timber 48oz Apple
Grapevine Goodness 48oz
Grapevine Goodness 12oz Conc
Grapevine Goodness 64oz
Grapevine Goodness 24oz
Savory Summer 12oz Raspberry
Savory Summer 16oz Raspberry
Lasting Lemon 12oz Lemon
Lasting Lemon 16oz Lemon
Citrus Plantation 36oz Lemon
Citrus Plantation 24oz Lemon
Citrus Plantation 36oz Lemon/Li
Citrus Plantation 24oz Lemon/Li
Regatta 24oz Raspberry
Regatta 24oz Raspberry/Grape
Regatta 36oz Raspberry/Grape
Regatta 36oz Raspberry
Regatta 48oz Raspberry
Warm Springs 24oz Strawberry
Warm Springs 24oz Cherry
Warm Springs 24oz Blueberry
Courtyard
Summer Delight
Fall Berry Festival
Timeless Timber
Grapevine Goodness
Savory Summer

EXHIBIT

Citrus Plantation

Raspberry Regatta

Warm Springs

Fruit Juice

Iced Tea

Seltzer Water

TOTAL PRODUCT

... This example displays the members of the **PRODUCT** dimension

... in the **JUICE** database in the range 1-20:

USE JUICE

EXHIBIT DIMENSION PRODUCT RANGE 1-20

... The output of this command would be:

Courtyard 12oz Conc.

Courtyard 12oz

Courtyard 16oz

Courtyard 36oz

Courtyard 48oz

Courtyard 64oz

Summer Delight 12oz Grape Conc.

Summer Delight 48oz Grape

Fall Berry 24oz Cran/Apple

Fall Berry 48oz Cran/Apple

Fall Berry 64oz Cran/Grape Conc

Fall Berry 24oz Cran/Grape

Fall Berry 48oz Cran/Grape

Timeless Timber 12oz Apple

Timeless Timber 36oz Apple

Timeless Timber 48oz Apple

Grapevine Goodness 48oz

Grapevine Goodness 12oz Conc

Grapevine Goodness 64oz

Grapevine Goodness 24oz

... This example displays the members of the **PRODUCT** dimension in the **JUICE** database in the

... range 1-20 sorted in reverse order, that is, from 20 to 1:

USE JUICE

EXHIBIT DIMENSION PRODUCT SORT REVERSE RANGE 1-20

... The output of this command would be:

Warm Springs 24oz Strawberry

Warm Springs 24oz Cherry

Warm Springs 24oz Blueberry

Warm Springs

TOTAL PRODUCT

Timeless Timber 48oz Apple

Timeless Timber 36oz Apple

Timeless Timber 12oz Apple

Timeless Timber

Summer Delight 48oz Grape

Summer Delight 12oz Grape Conc.

Summer Delight

Seltzer Water

Savory Summer 16oz Raspberry

Savory Summer 12oz Raspberry

Savory Summer

Regatta 48oz Raspberry

Regatta 36oz Raspberry/Grape

Regatta 36oz Raspberry

Regatta 24oz Raspberry/Grape

6.70.37 EXHIBIT MONTHS

Displays the short names of months, using the currently selected language.

Example

... This example displays the short month names in English:

EXHIBIT MONTHS

... The output of this command would be:

Jan

Feb

Mar

Apr

May

Jun

Jul

Aug

Sep

Oct

Nov

Dec

6.70.38 EXHIBIT OEM

Displays the contents of the OEM;TBDB document set, if it exists.

EXHIBIT

Example

... This example displays the OEM;TBDB set:

USE JUICE

EXHIBIT OEM

6.70.39 EXHIBIT PERIOD variable

Displays the time periods for the variable specified by the <variable> parameter.

Example

... This example displays the time periods for the variable SALES in the JUICE database:

USE JUICE

EXHIBIT PERIOD SALES

... The output of this command would be:

Jan 96

Feb 96

Mar 96

Apr 96

May 96

Jun 96

Jul 96

Aug 96

Sep 96

Oct 96

Nov 96

Dec 96

Jan 97

Feb 97

Mar 97

Apr 97

May 97

Jun 97

6.70.40 EXHIBIT periodicity PERIOD daterange 'dateformat' [LOCALE]

Parameter	Description
<periodicity>	Displays the data at one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, or hourly.
PERIOD	Displays the data at the specified periodicity and within the specified date range.
<daterange>	A range of dates separated by a dash (-). For example, 99/12/3 - 00/01/31 is the period from December 3, 1999 through January 31, 2000, and 99 - 00 is all of 1999 and 2000.

'<dateformat>'	Any reserved letters, and any text you want to appear in the date format. The reserved letters are preceded by a percent sign (%), and are as follows: y (year), f (fiscal year), m (month), d (day), h (hour, 12-hour clock), t (hour, 24-hour clock), s (am or pm), S (AM or PM), mi (minute), n (period number), and w (weekday name). The lowercase letters y, f, m, d, h, t, mi, and w indicate the starting year, month, day, and so on. The uppercase letters Y, F, M, D, H, T, MI, and W indicate the ending year, month, day, and so on.
LOCALE	Displays the months in the language in use by the operating system for French or German. If this option is not specified, the months are displayed in English.

Examples

... This example displays the monthly periods for 1997:

USE JUICE

EXHIBIT MONTHLY PERIOD 1997

... The output of this command would be:

Jan 97

Feb 97

Mar 97

Apr 97

May 97

Jun 97

Jul 97

Aug 97

Sep 97

Oct 97

Nov 97

Dec 97

... This example displays the quarterly periods for 1997:

USE JUICE

EXHIBIT QUARTERLY PERIOD 1997

... The output of this command would be:

Jan 97 - Mar 97

Apr 97 - Jun 97

Jul 97 - Sep 97

Oct 97 - Dec 97

Examples

To display the month in alphabetic character format, use a single M. For example, '%2D %3M %2Y' gives 31 Jan 99.

To display the month in numeric format, use MM. For example, '%2D %2MM %2Y' gives 31 01 99.

To specify a single digit day or month, use %D or %MM without a number preceding the letters. For example, '%D-%3M-%2Y' gives 1-Jan-99.

To display a 2-digit year, use %2Y. For example, '%D/%2MM/%2Y' gives 31/01/99.

To display a 4-digit year, use %Y without a number preceding the Y. For example, '%Y' gives 1999.

EXHIBIT

To display the full weekday name, use %W.

To abbreviate the weekday name, use a number with W. For example, '%3w %2d %3m %2y' gives Tue 01 Nov 99.

This table shows the default date formats for the Application Server default periodicities:

Periodicity	Default output	Default date format
avgtyd	1999 Avg	%Y Avg
bimonthly	Jan-Feb 00	%3m-%3M %2y
biweekly	01 Jan-14 Jan 00	%2d %3m-%2D %3M %2Y
daily	01 Jan 00	%2d %3m %2y
hourly	12am 30 Jan	%h%s %d %m
lunar	28 Jan 00	%2D %3M %2Y
monthly	Jan 00	%3m %2y
mtd	Jan 00 Mtd	%3M %2Y Mtd
mytd	Jan 00 Ytd	%3M %2Y Ytd
qtd	Jan-Mar Qtd	%3m-%3M Qtd
quarterly	Jan-Mar 00	%3m-%3M %2Y
rmonthly	31/01/00	%D/%2MM/%2Y
rquarterly	Jan-Mar 00	%3m-%3M %2Y
ryearly	2000	%Y
semiannual	Jan-Jun 00	%3m-%3M %2Y
weekly	07 Jan 00	%2D %3M %2Y
wtd	07 Jan 00	%2D %3M %2Y
yearly	2000	%Y
ytd	2000 Ytd	%Y Ytd

6.70.41 EXHIBIT PERIODICITY

Displays the shortest or longest or both periodicities of measures in the dimensional model. **Note:** When using this command, make sure you type at least the first seven letters of the word PERIODICITY. Otherwise Application Server will recognize it as the EXHIBIT PERIOD command.

The full syntax for this command is:

EXHIBIT PERIODICITY

{SHORTEST | LONGEST | BOTH} [SELECTED] [NOVIRTUAL] [CONSTANT] [EXCLUDE]
{SET}

Parameter	Description
SHORTEST	Displays the shortest periodicity of all the measures in the dimensional model. For example, if the measures in the dimensional model have a yearly, quarterly, monthly, and weekly periodicity, the output "Wee" will be displayed for weekly.
LONGEST	Displays the longest periodicity of all the measures in the dimensional model. For example, if the measures in the dimensional model have a yearly, quarterly, monthly, and weekly periodicity, the output "Yea" will be displayed for yearly.
BOTH	Displays the shortest and longest periodicity of all the measures in the dimensional model. The output appears as two strings separated by a TAB character. For example, if the measures in the dimensional model have a yearly, quarterly, monthly, and weekly periodicity, the output will appear as "Wee" "Yea".
SELECTED	Displays the shortest, longest, or both shortest and longest periodicity for only selected measures.

NOVIRTUAL	Displays the shortest, longest, or both shortest and longest periodicity for all measures that are not virtual.
CONTANTS	Displays the shortest, longest, or both shortest and longest periodicity for constant measures including attribute variables.
EXCLUDE	Displays the shortest, longest, or both shortest and longest periodicity for all measures except those listed in an EXCLUDEVAR document.
SET	Displays the periodicity corresponding to the current SET <periodicity> command. If no periodicity is set, you will get "DEFAULT".

Possible periodicity output:

"Bie"	Biennial
"Yea"	yearly
"Rye"	Ryearly
"Ytd"	Year to date
"Avg"	Average Year to date
"Sem"	Semi-annual
"Qua"	Quarterly
"Rqu"	Rquarterly
"Qtd"	Quarter to date
"Aqt"	Average quarter to date
"Bim"	BiMonthly
"Mon"	Monthly
"Rmo"	Rmonthly
"Mtd"	Month to date
"Amt"	Average month to date
"Myt"	Mytd
"Lun"	Lunar
"Ltd"	Lunar to date
"Pbi"	PBIWeekly
"Biw"	BIWeekly
"Wtd"	Week to date
"Wee"	Weekly
"Bus"	BSDaily
"Dai"	Daily
"Shi"	Shift
"Hsh"	Half Shift
"Hourly"	Hourly
"Qho"	Qhourly
"Con"	Constant

6.70.42 EXHIBIT PROCESSID

Displays the system process ID.

6.70.43 EXHIBIT SDIMENSION

**[dimension] [EXCLUDE] [FULL]
[FORMAT '%s text']**

Lists all structural dimensions.

Parameter	Description
<dimension>	Specifies the dimension about which you want to display information. Omit this parameter to display information about all structural dimensions.
EXCLUDE	Checks for the existence of a document set called EXCLUDEDIM in the Use database; the document specifies a list of dimensions and attributes that should

EXHIBIT

be excluded from the EXHIBIT list. If you do not specify EXCLUDE, details of all structural dimensions are listed.

Note: If a dimension is listed in EXCLUDEDIM, its attributes will still be shown by EXHIBIT SDIMENSION. To exclude attributes, as well as dimensions, from the EXHIBIT list, you must specify them in EXCLUDEDIM.

FULL

Displays full details for the dimension or dimensions in tab-separated format in the following order:

Dimension name

Maximum number of input members

Actual number of inputs

Maximum number of outputs

Actual number of outputs

Maximum number of result members (0 or 1)

Actual number of result members (0 or 1)

Number of levels

Comma-separated list showing the number of members in each level (ordered from input to result)

Number of selected members

Maximum number of User-Defined Hierarchies

Actual number of User-Defined Hierarchies

FORMAT '%s <text>'

Display the specified text with the dimensions appended in the %s location. This allows you to quickly build a set of similar commands. For example:

EXHIBIT SDIM FORMAT 'SEL %s RESULT'

SEL CHANNEL RESULT

SEL CUSTOMER RESULT

SEL PRODUCT RESULT

Example

... This example displays the structural dimensions of the JUICE database:

USE JUICE

EXHIBIT SDIMENSION

... The output of this command would be:

CHANNEL

CUSTOMER

PRODUCT

... This example displays full details of the PRODUCT dimension:

USE JUICE

EXHIBIT SDIMENSION PRODUCT FULL

... The output of this command would be:

PRODUCT	37	36	18	12	1	1	4	36,9,3,1 49	5	0
---------	----	----	----	----	---	---	---	-------------	---	---

6.70.44 EXHIBIT QUERY [VARIABLES] { DETAILS | SETTINGS } [LINKID < LinkId>] [CUBE <cube>] [<variable>]

Use

This command dumps out a subset of the information from the SAP VARIABLES rowset that is cached in Application Server. The output appears in tab-separated form.

Syntax

EXHIBIT QUERY [VARIABLES] {DETAILS|SETTINGS} [LINKID <LinkId>] [CUBE <cube>]
[<variable>]

Parameter	Description
DETAILS	Supplies most information from the rowset with any current setting for a Variable as the last field: LINKID CUBE_NAME VARIABLE_NAME VARIABLE_CAPTION VARIABLE_TYPE DATA_TYPE CHARACTER_MAXIMUM_LENGTH PROCESSING_TYPE SELECTION_TYPE ENTRY_TYPE REFERENCE_DIMENSION REFERENCE_HIERARCHY DEFAULT_LOW DEFAULT_HIGH DEFAULT_LOW_CAPTION DEFAULT_HIGH_CAPTION DESCRIPTION CURRENT_SETTING
SETTINGS	Displays a minimal set of information with any current setting added as the last field: LINKID CUBE_NAME VARIABLE_NAME VARIABLE_CAPTION CURRENT_SETTING
LINKID <LinkId>	Enter a specific LinkId if the Application Server database contains imported information from more than one Query and you want Query Variable information for just one Query Cube. If the Application Server database contains only imported information from a single Query, you do not need to supply a specific LinkId.
CUBE <cube>	Enter a specific cube if the Application Server database contains imported information from more than one Query and you want Query Variable information for just one Query Cube. If the Application Server database contains only imported information from a single Query, you do not need to supply a specific cube.

6.70.45 EXHIBIT SECURITY {dimension | VARIABLES | LEVELS |

VIEWERFORMAT | FORMAT '%s text %d' | pattern } [EXCLUDE] [FULL]

Use

Identifies whether the item has security applied to it or not. When the EXHIBIT SECURITY command is executed, Application Server reviews any Security procedure that exists to see if this dimension has security applied in an INDEX USER-CASE-ENDINDEX statement. If the item is specified in the statement, this command displays a 1. If the item is not specified in the statement, this command displays a 0.

Syntax

EXHIBIT SECURITY

```
{ <dimension> | VARIABLES | LEVELS | VIEWERFORMAT | FORMAT '%s <text> %d' |
  <pattern> }
[EXCLUDE] [FULL]
```

Parameter	Description
<dimension>	Displays a 0 if this dimension does not have security applied in a Security procedure. Displays a 1 if this dimension does have security applied.
VARIABLES	Displays a 0 if any of the variables do not have security applied in a Security procedure. Displays a 1 if any of the variables do have security applied.
LEVELS	Shows three columns of information: a count of the number of hierarchies in a dimension, the dimension name, and a 0 or 1 for whether the dimension has security applied in a Security procedure. It displays a 0 if the dimension does not have security applied, and displays a 1 if the dimension does have security applied. This example output shows that all dimensions have one hierarchy, and the Product dimension has security applied: EXHIBIT SECURITY LEVELS 1 CHANNEL 0 1 CUSTOMER 0 1 PRODUCT 1
FULL	Displays the following tab-separated columns of information for each dimension: Column Description 1 Dimension name 2 Maximum number of input members 3 Actual number of inputs 4 Maximum number of outputs 5 Actual number of outputs 6 Maximum number of result members (0 or 1) 7 Actual number of result members (0 or 1) 8 Number of levels 9 Comma-separated list showing the number of members in each level (ordered from input to result) 10 Number of selected members 11 Maximum number of User-Defined Hierarchies 12 Actual number of User-Defined Hierarchies 13 Whether the dimension has security applied. (0 if no security applied, 1 if security is applied).

VIEWERFORMAT	Displays the dimension, then its default member that would be displayed in any Viewer dialog box, and then either a 0 or 1 depending on whether the dimension has security applied. It displays a [0] if the dimension does not have security applied in a Security procedure, and displays a [1] if the dimension does have security applied.
EXCLUDE	<p>Checks for the existence of a document set called EXCLUDEDIM in the Use database; the document specifies a list of dimensions that should be excluded from the EXHIBIT list. The resulting list shows the dimensions not excluded from the EXCLUDEDIM document set, and also includes a 0 or 1 for whether the dimension has security applied.</p> <p>For example, assume an EXCLUDEDIM document contains the Channel dimension and State attribute. The result list shows all dimensions and attributes except Channel and State, and includes whether there is security applied to that dimension:</p> <pre> EXHIBIT SECURITY EXCLUDE CONTAINER 0 COT 0 CUSTOMER 0 FLAVOR 0 PARENT 0 PRODUCT 1 SIZE 0 </pre>
FORMAT '%s <text> %d'	<p>Displays each dimension, the text to append to each dimension, and also displays a 0 or 1 for whether the dimension has security applied.</p> <p>For example:</p> <pre> EXHIBIT SECURITY FORMAT '%s security %d' CHANNEL security 0 CONTAINER security 0 COT security 0 CUSTOMER security 0 FLAVOR security 0 PARENT security 0 PRODUCT security 1 SIZE security 0 STATE security 0 </pre>
<pattern>	<p>Specifies a wildcard pattern that displays the dimensions that match the pattern and whether the dimensions have security applied. Use an asterisk (*) in a name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name.</p> <p>The output displays the dimension name and then a 0 if the dimension does not have security applied, and displays a 1 if the dimension does have security applied.</p> <p>In this example, the c* wildcard pattern is used to return the following results:</p> <pre> EXHIBIT SECURITY c* CHANNEL 0 CONTAINER 0 COT 0 CUSTOMER 0 EXHIBIT SECURITY p* PARENT 0 PRODUCT 1 </pre>

6.70.46 EXHIBIT settype [pattern] [FROM database] FULL

Displays information about document sets in a database.

Parameter	Description
<settype>	Specifies the type of set — binary, dimension, document, logic, procedure, report, synonym, or time.
<pattern>	Specifies a wildcard pattern that set names must match. Omit this parameter to display all sets of the type <settype>.
FROM <database>	Specifies which database to search for sets. The specified database must be attached. Omit this parameter to display sets in the Use database.
FULL	Displays detailed information about the sets in tab-separated format as follows: Name Number of lines Number of database blocks occupied by the set Date and time of creation Date and time last modified

Examples

... This example displays a list of procedures in the JUICE database:

USE JUICE

EXHIBIT PROCEDURE FROM JUICE

... The output of this command would be:

CUSTOMER

MBJUICE

PRODUCT

... This example displays a list of reports in the DEMOMH database:

USE DEMOMH

EXHIBIT REPORT FROM DEMOMH

... The output of this command would be:

EXAMPLE

REGIONVAR

REPVAR

REPVAR2

TRENDVAR

6.70.47 EXHIBIT STRUCTURE

This command displays the following information:

Use database name.

Name of the time set being used.

ACROSS and DOWN list of dimensions and variables.

Whether SET SHORT or SET LONG is in effect.

The time period specified by a SET PERIOD command.

The latest time period specified by a SET LATEST command.

Selected variables.

Dimension names, the selected members, and the top member of each dimension.

Example

... This example displays the structure of the JUICE database:

USE JUICE

EXHIBIT STRUCTURE

... The output of this command would be:

USE JUICE

SET LONG

SET PERIOD DEFAULT

SET TIME NONE

SET LATEST Mon 31 Oct 1996

SEL VARIABLES 'Container'

ACROSS TIME DOWN VARIABLES

...DIM CHANNEL TOTAL CHANNEL

SEL CHANNEL #3

...DIM CUSTOMER 7 Eleven - Nashua, NH

SEL CUSTOMER #1

...DIM PRODUCT Courtyard 12oz Conc.

SEL PRODUCT #1

6.70.48 EXHIBIT TOPS ...

The syntax for this command is:

EXHIBIT TOPS [**FORMAT** <formatstring>] [**POS**] [**BOTH**] [**SELECT**] [<setname>] [**OVER**]

Displays the top member for each dimension in the format:

Dimension \t Top Member

Parameter	Description
TOPS	Lists the dimension's top member, which normally is the result. If the dimension has no result member, Application Server displays the first member from the highest output level. If the dimension has no outputs and no result, the first input member is displayed. Note: This command produces the same result as EXHIBIT DIMENSION <dimension> TOP.
FORMAT <formatstring>	Formats the output according to the specified <formatstring>. You can embed other text and punctuation in the format string, and must specify a "%s" for every token in the output string. Note: If SET SHORT is in operation, the short names include an underscore. If SET LONG is in operation, the underscore is removed from the name.
POS	Displays the position of the top member within the dimension in the format: Dimension \t Hash Number \t Top Member

EXHIBIT

BOTH	Displays the short and long names for the top member in the format: Dimension \t Short Name \t Long Name
SELECT	Displays and selects the top member.
<setname>	Creates a procedure that selects the specified top member.
OVER	Specifies that any existing set with the same name as <setname> should be overwritten by the new set.

Examples

... This example displays and selects the top members of all dimensions in the JUICE database:

USE JUICE

EXHIBIT TOPS

... The output of this command would be:

```
CHANNEL      TOTAL CHANNEL
CUSTOMER     TOTAL CUSTOMER
PRODUCT      TOTAL PRODUCT
```

... This example displays and selects the top members of all dimensions in the JUICE database, and

... formats the output using the string "%s %s":

USE JUICE

EXHIBIT TOPS FORMAT "%s %s"

... The output of this command would be:

```
CHANNEL 'TOTAL CHANNEL'
CUSTOMER 'TOTAL CUSTOMER'
PRODUCT 'TOTAL PRODUCT'
```

6.70.49 EXHIBIT TRANSACTION

Displays a count of the commands executed since the start of the session.

Example

... This example displays the number of commands executed in the current session:

EXHIBIT TRANSACTION

... The output of this command would be:

15

6.70.50 EXHIBIT USER [username] [FULL]

Displays the names of users in MASTERDB.

Parameter	Description
<username>	Specifies the name of a user if it exists in MASTERDB. Omit this parameter to display all users in MASTERDB.
FULL	Displays the properties of the user(s). Note: This option is available only to Supervisors. Returns a tab-separated string in the following format: Name \tSession No \tPassword \tSecurity \tStatus \tLast Login \tLast Paused \tExpiration Date \tDefault Model \tDefault Usage \tWork database \tBlocks \tBlock Size \tLibrary database \tBuffers \tMemory \tMax Logins \tGroups

Example

... This example displays details of all the users in MASTERDB:

EXHIBIT USER FULL

... The output of this command would be:

```
ADMIN 1      Supervisor   Logged in      2000/04/03 07:55:19      2000/04/02
      17:29:58      JUICE Exclusive   WKADMIN      10000 8192      TBDB 0
      2000      1      TEST,MANAGE

INITIAL 1    Supervisor   Enabled2000/04/02 15:28:10      INITIAL
      Read-only      WKINIT 10000 4096 TBDB 0      2000 1

SUPER 1      Supervisor   Logged in      2000/02/24 10:35:05
      JUICE Read-only      WKSUPER      10000 8192 TBDB 0      2000 1

SUPERVISOR 1 ***** Supervisor   Enabled2000/04/02 14:42:42
      Exclusive SUPENVDB 200 8192 TBDB 0      3000 1
      TEST,MANAGE
```

... This example displays details of the ADMIN user:

EXHIBIT USER ADMIN FULL

... The output of this command would be:

```
ADMIN 1      Supervisor   Logged in      2000/04/03 07:55:19      2000/04/02
      17:29:58      JUICE Exclusive   WKADMIN      10000 8192      TBDB 0
      2000      1      TEST,MANAGE
```

6.70.51 EXHIBIT VARIABLE [ONLY] [JUST] BELOW <virtual_variable>

Parameter	Description
ONLY BELOW	Lists all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also listed. The specified virtual variable is not included in the list.
JUST BELOW	Lists the specified virtual variable and all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is listed, but the variables that make up that virtual variable are <i>not</i> listed.
BELOW	Lists the specified virtual variable and all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also listed.
ONLY JUST BELOW	Lists all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is listed, but the variables that make up that virtual variable are <i>not</i> listed. The specified virtual variable is not included in the list.
<virtual_variable>	Specifies the virtual variable whose variables and virtual variable hierarchies you want to display.

6.70.52 EXHIBIT VARIABLE [<variable> [<value>]] [BOTH] [SORT [REVERSE [LONG]] [MDXNAME] [RANGE <integer> - <integer>]]

Parameter	Description
VARIABLE	Displays the names of all variables, virtual variables, and attributes.

EXHIBIT

<variable>	Displays information for the specified variable.
<value>	For undimensioned constant variables, displays the value of the variable according to the specified variable format.
LIKE	Displays the names of all variables similar to the specified variable based on dimensions and periodicity.
BOTH	Displays both short and long names for the specified variable. The BOTH keyword ignores the SET SHORT/LONG setting, as appropriate.
SORT	Sorts the output from the EXHIBIT command.
REVERSE	Sorts in reverse order (last to first).
LONG	Sorts the long names of the variables. When using the SORT LONG keywords, you must also use the BOTH keyword, which displays both short and long names. For example, EXHIBIT VARIABLE BOTH SORT LONG.
MDXNAME	When displaying data from an SAP NetWeaver BI InfoCube, this keyword exhibits the MDX variable name immediately after the Application Server short and/or long name in the output.
RANGE	Limits the sort to the specified range of variables between and including the first integer and the last integer.

6.70.53 EXHIBIT VARIABLE <variable> CONVERT [BEFORE| AFTER]

Parameter	Description
VARIABLE	Displays the names of all variables, virtual variables, and attributes.
<variable>	Displays information for the specified variable.
CONVERT BEFORE AFTER	Returns information about the SET VARIABLES CONVERT settings. It is only relevant for virtual measures. For example, EXHIBIT VARIABLE <variable> CONVERT gives the result: VARIABLE= <variable> CONVERT= BEFORE.

6.70.54 EXHIBIT VARIABLE FROM { <report> | <logic> | <virtual_variable> } [RECURSE] [EXCLUDE] [SORT [REVERSE]]

Parameter	Description
FROM	Displays variables from the specified report or logic set or virtual variable.
<report>	Displays variables from the specified report.
<logic>	Displays variables from the specified logic set.
<virtual_variable>	Displays variables from the specified virtual variable.
RECURSE	Displays the variables referenced in the logic, report, or virtual variable. If a virtual variable is used in the specified logic, report, or virtual variable, then RECURSE also displays the variables that compose that virtual variable.
SORT	Sorts the variables in alphabetical order from a to z.
REVERSE	Reverses the sort order of variables so they are listed from z to a.
EXCLUDE	Displays the specified variables, and omits hidden variables.

6.70.55 EXHIBIT VARIABLE LIKE

<variable> [value] [EXCLUDE]

Parameter	Description
LIKE	Displays the names of all variables similar to the specified variable based on dimensions and periodicity.
<variable>	Displays information for the specified variable.
<value>	For undimensioned constant variables, displays the value of the variable according to the specified variable format.
EXCLUDE	Displays the names of all variables similar to the specified variable based on dimensions and periodicity, and excludes any variables listed in the Excludevar document, if one exists. Note: You cannot use the EXCLUDE keyword when using EXHIBIT for a single measure.

Example

... This displays details of all variables in the JUICE database that are similar to the SALES variable:

USE JUICE

EXHIBIT VARIABLE LIKE SALES

... The output of this command would be:

Costs

Margin

Quota

Sales

Units

Variance

... This example displays details of the SALES variable:

USE JUICE

EXHIBIT VARIABLE SALES

... The output of this command would be:

VARIABLE=SALES

PERIODICITY=MON

LABEL=Sales

BYTES=4

NOCONSOLIDATE=YES

DIMENSIONS=CHANNEL, CUSTOMER, PRODUCT

NOBS=18

RANGE=Jan 96 - Jun 97

WIDTH=15

DECIMALLOCALE=No

DECIMALS=2

POINTLOCALE=Yes

POINT=.

EXHIBIT

COMMALocale=Yes

COMMA=,

NEGATIVE=-

AUTO=YES

ZERO=

MISSING=-

... This example creates an Excludevar doc and exhibits all virtual variables and all variables

... not included in the Excludevar:

USE JUICE

DOCUMENT EXCLUDEVAR

Costs

Units

Margin

EXHIBIT VARIABLES LIKE SALES EXCLUDE

... The output of this command would be:

Margin

Quota

Sales

Variance

6.70.56 EXHIBIT VARIABLES [EXCLUDE]

Lists all variables, virtual variables, and attributes.

Parameter	Description
EXCLUDE	Lists all variables, virtual variables, and attributes in the current Use database, and excludes any variables listed in the Excludevar document, if one exists. If you specify EXHIBIT VARIABLE EXCLUDE, details of all variables not included in the EXCLUDEVAR document are listed.

Example

... This example lists all variables and attributes in the DEMOMH database:

USE DEMOMH

EXHIBIT VARIABLES

... The output of this command would be:

Cost of goods

Margin

Margin %

Price

Sales

Standard Cost

Units

... This example creates an Excludevar doc and exhibits all attributes and virtual variables, and

... any variables not included in the Excludevar:

USE JUICE

DOCUMENT EXCLUDEVAR

Costs

Units

Margin

EXHIBIT VARIABLES EXCLUDE

... The output of this command would be:

Container

Class of Trade

Flavor

Margin

Parent

Quota

Sales

Size

State

Variance

6.70.57 EXHIBIT VVARIABLE [vvarname] [BOTH] [EXCLUDE]

Lists the virtual variable names.

Parameter	Description
<vvarname>	Specifies the virtual variable. If you do not specify a virtual variable, all virtual variables in the database are listed.
BOTH	Displays both the long and the short name of the virtual variable.
EXCLUDE	Lists the virtual variables in the database, and excludes any virtual variables listed in the Excludevar document, if one exists. Note: You cannot use the EXCLUDE keyword when using EXHIBIT for a single virtual variable.

... This example lists the virtual variables in the JUICE database:

USE JUICE

EXHIBIT VVARIABLE

... The output of this command would be:

Margin

Variance

... This displays the long and short names of the MARGIN virtual variable in the JUICE database:

USE JUICE

EXHIBIT VVARIABLE MARGIN BOTH

... The output of this command would be:

MARGIN Margin

... This example creates an Excludevar doc and exhibits all attributes and virtual variables, and

... any variables not included in the Excludevar:

EXHIBIT

USE JUICE

DOCUMENT EXCLUDEVAR

Costs

Units

Margin

EXHIBIT VVARIABLES BOTH EXCLUDE

... The output of this command would be:

VARIANCE Variance

6.70.58 EXHIBIT VVARIABLE

[<vvarname>] { [options] [LOGIC] [EXCLUDE] | COMPILE] | FORMAT '<%s text>'}

Displays virtual variable properties.

Parameter	Description
<vvarname>	Specifies the virtual variable. If you do not specify a virtual variable, all virtual variables in the database are listed.
<options>	Specifies the usual options available with measures and variables.
LOGIC	Displays the virtual variable name followed by the formula for the virtual variable.
EXCLUDE	Displays a list of all virtual variable properties, and excludes any virtual variables listed in the Excludevar document, if one exists.
FORMAT '<text %s>'	Generates a command line for virtual variables based on the text string. For example, you can use the following command to recompile hidden logic sets used by virtual variables: EXHIBIT VVARIABLE FORMAT "Compile Logic '% % %s' "
COMPILE	Generates a command line to recompile the hidden virtual logic in virtual variables. This keyword always uses the short name of the virtual variable regardless of any SET SHORT/LONG setting.

Example

... This displays the details of the virtual variables in the JUICE database, including the formulas:

USE JUICE

EXHIBIT VVARIABLE LOGIC

... The output of this command would be:

Margin Sales - Costs

Variance Sales - Quota

Rules

Hierarchies are numbered by the order in which they were defined in the CONSTRUCT command. The first HIERARCHY statement creates a hierarchy whose number is 1. The second HIERARCHY statement creates a hierarchy numbered 2.

If a dimension does not have multiple hierarchies, Application Server considers it to have one hierarchy.

A dimension's default hierarchy is the one defined in the first HIERARCHY statement of the CONSTRUCT command. You can set a different hierarchy with the SET command.

6.71 EXIT

Use

EXIT temporarily suspends an Application Server session. When you enter EXIT, Application Server executes a procedure named EXIT (located in the database) before exiting. When you enter EXIT CLEAR, Application Server executes procedures named EXIT and EXITCLEAR (if found in the database) before exiting. During an EXIT CLEAR, any across/downs, set periods, periodicities, variable, and dimension selections are removed.

Note: When MAXLOGINS>1, a new Work database is always created when you log in or switch to a user. Any previous Work database is not used on a restart or a SWITCH command, so the current Work database is considered discardable.

Syntax

EXIT [KEEP] [CLEAR]

Parameter	Description
-----------	-------------

EXIT

	Logged in with MAXLOGIN>1	Logged in with MAXLOGIN=1
SWITCH WORKDB used in session	Clears the previous Work databases and the current Work database.	Clears the previous Work databases and the current Work database.
SWITCH WORKDB not used in session	Clears the current Work database.	Keeps the current Work database.

EXIT KEEP

	Logged in with MAXLOGIN>1	Logged in with MAXLOGIN=1
SWITCH WORKDB used in session	Keeps the previous Work databases and clears the current Work database.	Keeps the previous Work databases and the current Work database.
SWITCH WORKDB not used in session	Clears the current Work database.	Keeps the current Work database.

EXIT CLEAR

	Logged in with MAXLOGIN>1	Logged in with MAXLOGIN=1
SWITCH WORKDB used in session	Clears the previous Work databases and the current Work database.	Clears the previous Work databases and the current Work database.
SWITCH WORKDB not used in session	Clears the current Work database.	Clears the current Work database.

EXIT KEEP CLEAR

	Logged in with MAXLOGIN>1	Logged in with MAXLOGIN=1
SWITCH WORKDB used in session	Keeps the previous Work databases and clears the current Work database.	Keeps the previous Work databases and clears the current Work database.

EXP()

SWITCH WORKDB not used in session	Clears the current Work database.	Keeps the current Work database.
--	--------------------------------------	-------------------------------------

Example

...This returns you to the host operating system. When you log in to Application Server again, data ... in the temporary workspace from previous login will be in the same state as when you exited.

EXIT

...This example returns you to the host operating system. Application Server deletes all the ...data from your temporary workspace.

EXIT CLEAR

6.72 EXP()

Use

EXP returns the value of $e(2.71828...)$ raised to the specified power.

Syntax

CALCULATE <result> = EXP(<power>)

Parameter	Description
<result>	Name of the result variable.
<power>	Power of e.

6.73 EXTEND

Syntax

EXTEND <database> <blocks>

Use

EXTEND increases the size of a database.

Parameter	Description
<database>	Name of the database to be extended.
<blocks>	Number of blocks to add to the database. Note: Application Server does not preallocate blocks; it uses them as needed. This means that it does not set aside a fixed number of blocks of disk space when you specify a database. This block value is the size that the database is not allowed to exceed. You should check the availability of disk space when you specify a block size and monitor availability when using the database. For example, you could specify a database of a certain size, slowly fill that database, and run out of disk space before you reach your specified maximum.

EXTEND - Rules

You can only extend a database that you have attached as Update Exclusive.

When you enter an EXTEND command for a partitioned database, Application Server compares the new size with the partition information in MASTERDB. If you extend the database to a size that is larger than the size of the partitions, you get an error message. You can then add partitions using the CHANGE DATABASE command.

Example 1: EXTEND

...This example increases the size of the database BUDGET by 200 blocks:

EXTEND budget 200

Example 2: EXTEND

...This example detaches a database to increase the number of database partitions. Because the new ... database partitions exceed the database's BLOCKS specification, Application Server uses the ... database and then extends it so that the database size equals the increased partition sizes.

DETACH Accnt

CHANGE DATABASE Accnt ADD 2 PARTITIONS SIZES 40K, 20K

...The output of this command would be:

Warning: The Number of Blocks Specified in the PARTITION SIZES (120000) for Database ACCNT is Larger than the BLOCKS specified.

Database: ACCNT

Maxaccess: UPDATE Disposition: ENABLED

Current State: Available Protection Key: None

Last user: Last access:

Blksize: 8192 Observations 500

Maxblocks: 60000 Number Free: 59990

Database ACCNT Partition Information.

Number of Partitions: 5

Blocksize: 8192

Date Last Opened for Update: Thu Feb 9 15:34:10 2000

Date Last Closed After Update: Thu Feb 9 15:34:10 2000

Partition	Size	Name
1	20000	ACCNT01
2	20000	ACCNT02
3	20000	ACCNT03
4	20000	ACCNT04
5	20000	ACCNT05

USE Accnt

EXTEND Accnt 60000

SHOW DATABASE

...The output of this command would be:

USE Database: ACCNT Max. Obs: 500

	Access	Maxblocks	% Available	Mode
WKADMIN	UPDATE	2000	99%	EXCLUSIVE
TBDB	READ	400	2%	READONLY
ACCNT	UPDATE	120000	99%	EXCLUSIVE

EXTEND (Supervisor)

6.74 EXTEND (Supervisor)

Use

EXTEND is a Supervisor command that increases the size of MASTERDB by the number of blocks specified.

Syntax

EXTEND <integer>

Parameter	Description
<integer>	The number of blocks to add to MASTERDB. Note: If there is not enough disk space to add the specified number of blocks, Application Server lowers the integer value to an acceptable number of blocks.

Example

...This example increases the size of MASTERDB by 25 blocks:

EXTEND 25

6.75 FOOTNOTE-ENDFOOTNOTE (Report)

Use

FOOTNOTE-ENDFOOTNOTE is a Report command that defines a block of footnotes that appear at the bottom of each page of a report.

Syntax

```
FOOTNOTE
    .
    . <footnote specifications>
    .
ENDFOOTNOTE
```

Example

...The following block produces the following footnote at the bottom of each page:

```
FOOTNOTE
    TEXT 'Note: Consolidated figures do not include special charges.'
ENDFOOTNOTE
```

Note: Consolidated figures do not include special charges.

6.76 FOR

Use

FOR restricts the periods in a logic statement. It can appear to the right, left, or both sides of an equal (=) sign.

Syntax

```
FOR {<n> [ -<n> ] }
    {LATEST [PLUS <m> | MINUS <m> ] }
    {LAST [PLUS <m> | MINUS <m> ] }
```

Parameter	Description
<n>	A period number in the range: 1, 2, ..., LAST.
LAST	Indicates the period is the last period in the selected date range.
PLUS	Indicates the period <m> periods after the specified period.
<m>	Number of periods before or after the last or latest period.
LATEST	Indicates the period is the one defined as the latest period for which data is available.
MINUS	Indicates the period <m> periods before the specified period.

Rules

FOR always refers to the expression that precedes it.

FOR can only restrict execution within a date range; it cannot expand the range.

FOR to the right of an equal sign only restricts the basis for evaluating the expression. It has no effect on which result values Application Server changes.

FOR to the left of an equal sign only restricts the periods into which the results of the expression to the right of the equal sign are stored. It has no effect on the periods used to evaluate the expression.

Example

...The following statement performs a projection of the average sales during the period 1

...through latest into the future periods latest plus 1, latest plus 2, ..., last:

CALCULATE Sales FOR LATEST PLUS 1 - LAST = MEAN(Sales) FOR 1 - LATEST

6.77 FORCE (Supervisor)

Use

You can use the FORCE command to make minor corrections to partitioned databases to allow Application Server to open them. The FORCE command also does an exhaustive XRAY on the database to ensure that it is consistent. This process can be time consuming for larger databases.

Syntax

FORCE <name>

When To Use this Command

Partitioned databases consist of several files, which might be on different drives and be backed up and restored at different times. Application Server maintains consistency between the various files by using information in the control file and the first block of each partition. These files might become desynchronized in various ways, in which case an ordinary open fails, and an informative message displays.

Note: If you think the failure is essentially harmless (bad or missing control file, inconsistent open or close times), use the FORCE command to try to correct the error. Since database error can have many causes, success is not guaranteed. FORCE can be a very dangerous command, since it can force possibly inconsistent parts of a partitioned database to appear to be problem free. In this case, it is recommended that the database be rebuilt or restored from a backup.

Parameter	Description
<name>	The name of the failed partitioned database.

6.78 FREE (Supervisor)

Use

FREE is a Supervisor command that changes either a user's login state or the state of a database.

FREE USER releases a user. Application Server deletes the user's temporary database, and changes the user's login state in the MASTERDB record to Enabled. FREE USER also changes the current state of that user's Use database to Available. The user is not, however, logged off Application Server.

FREE DATABASE releases a database. The current state field in the record for that database in MASTERDB is changed to Available.

FREE is normally used to restore the use of databases or user logins. Abnormal termination of Application Server, for example, might leave the user's login state in MASTERDB marked as On, and the database disposition in MASTERDB marked as In Use. This prevents the user from logging in, and prevents anyone from using the database.

Syntax

FREE {USER <user>[<session>] | DATABASE <database> | * }

Parameter	Description
USER	Frees specified users and their Use database.
<user>	Name of the user.
<session>	The session number. This frees sessions other than the first for a multiple login. Use this for users defined with a MAXLOGINS greater than 1.
DATABASE	Frees the specified database.
<database>	Name of the database.
*	Forces all users to start a new session, even if they only logged out with an EXIT. This command would be used by a system administrator who had rolled out a new release under instructions to use this command from the installation notes. This option implies that the format of the Work database has changed.

Notes:

Never use the FREE command when someone else is logged in. The effect is to clean out MASTERDB, so if someone else is logged in, errors may occur.

If a user is in the Supervisor subsystem, and you enter FREE USER for that user, Application Server terminates the user's session and all work from that session will be lost.

Examples:

...This example changes the login state for the user John from On to Enabled,

...and changes the state of his Use database to AVAILABLE:

SHOW USER John

...The output of this command would be:

User: JOHN

Use Database: RESRCH Temp Database: DB220266

Blocks: 700 Blksize 2048

Buffers: 300 Library Database: TBDB

Security Level: 0 Login State: On

Last Login: Wed Apr 8 15:17:41 2000

FREE USER John

...The output of this command would be:

Database: RESRCH

Maxaccess: UPDATE Disposition: ENABLED

Current State: AVAILABLE Protection Key: None

Last user: John Last access: Wed Apr 8 16:48:17 2000

Blksize: 2048

Maxblocks 700 Number Free 458

User: JOHN

Use Database: RESRCH Temp Database: DB220266

Blocks: 700 Blksize 2048

Buffers: 300 Library Database: TBDB

Security Level: 0 Login State: Enabled

Last Login: Wed Apr 8 15:17:41 2000

...If your system crashes, database records in MASTERDB might indicate that a database in the ... system is in use even after the system becomes operational. This example shows the changes in ... the current state of the database FIN1 from IN USE (UPDATE) to AVAILABLE:

SHOW DATABASE fin1

...The output of this command would be:

Database: FIN1

Maxaccess: UPDATE Disposition: ENABLED

Current State: IN USE (UPDATE) Protection Key: None

Last user: FIN1 Last access: Sun Mar 8 08:08:08 2000

Blksize: 2048

Maxblocks 200 Number Free 97

FREE DATABASE fin1

...The output of this command would be:

Database: FIN1

Maxaccess: UPDATE Disposition: ENABLED

Current State: AVAILABLE Protection Key: None

Last user: FIN1 Last access: Sun Mar 8 08:08:08 2000

Blksize: 2048

Maxblocks 200 Number Free 97

...This example frees sessions other than the first for a multiple login.

...You would use this for a user defined with a MAXLOGINS greater than 1.

FREE USER MANYLOG(4)

6.79 GET ENVIRONMENT

Use

Use the GET ENVIRONMENT command to get the value of environment variables (external shell variables), such as DBHOME, DBPATH, or database path names and put them into PAS control variables.

HEADING-ENDHEADING (Report)

Syntax

```
GET ENVIRONMENT <shell_variable> <control_variable>
```

Parameter	Description
<shell_variable>	External shell variable whose value will be used for the PAS control variable. On UNIX, the <shell_variable> name is case-sensitive but on WinX it is not (the code always upper cases the <shell variable name>). On UNIX, the shell variables are real UNIX shell variables. On WinX the shell variables are entries in the [Windows] section of Isserver.ini. Note: If the external shell variable does not exist, then the control variable is set to an empty value. You can view control variables using the SHOW CONTROL command.
<control_variable>	Name of the PAS control variable that will have the value of the shell variable.

Example

...This is an example of using GET ENVIRONMENT on a WinX machine:

```
System> SHOW CONTROL
```

Control Variable	Value
-----	-----
LIBRARY DATABASE	TBDB
SID	Windows
WORKDATABASE	WKADMIN

```
System> GET ENVIRONMENT dbpath path1_cv
```

```
System> SHOW CONTROL
```

Control Variable	Value
-----	-----
LIBRARY DATABASE	TBDB
PATH1_CV	C:\Program Files\SAP BusinessObjects\Strategy Management\InternetPub\Procs;C:\Program Files\SAP BusinessObjects\Strategy Management\ApplicationServer\home;C:\Program Files\SAP BusinessObjects\Strategy Management\ApplicationServer\data
SID	Windows
WORKDATABASE	WKADMIN

6.80 HEADING-ENDHEADING (Report)

Use

HEADING-ENDHEADING is a Report command that defines a block of headings that appear at the top of each page of a report.

Note: Only one heading block can occur in a report.

Syntax

```
HEADING
    .
    . <heading specifications>
    .
ENDHEADING
```


Example

...The following block:

```
HEADING
  TITLE 'KMS International'
  TITLE '2000 Profit And Loss Report'
  UTEXT
ENDHEADING
```

...produces the following heading at the top of each page:

```
      KMS International
2000 Profit And Loss Report
-----
```

6.81 HIERARCHY (Dimension)

Use

HIERARCHY is a Dimension command that defines the members and levels that comprise a particular hierarchy level within a dimension. It also allows multiple consolidation structures that stem from a central root of inputs. Use hierarchies when the input members are consolidated into more than one output member. When a dimension has hierarchies, members can have more than one path for rolling up into output levels. If a dimension has only one hierarchy, no HIERARCHY statement is necessary.

Syntax

```
HIERARCHY <hierarchy>
      LEVEL <field> [ ,...,<field> ]
```

Parameter	Description
<hierarchy>	Name of the hierarchy as defined in the CONSTRUCT <dimension> HIERARCHY <hierarchy> command. You can have up to eight HIERARCHY statements in a dimension. Note: The default hierarchy is the one defined in the first HIERARCHY statement. You can set a different hierarchy using the SET command.
LEVEL <field>	One or more field names, separated by commas. Each field represents a level of the hierarchy. The first <field> represents input members, the next <field> represents the first level of output members, and so on. Member names cannot be more than 24 alphanumeric characters.

Example

...This example shows two HIERARCHY statements in the dimension set. The first hierarchy, called ... REGIONS, contains two levels, Store and Sales_Region. The second hierarchy, called STATES, ... contains three levels, Store, State, and Geo_Region:

```
HIERARCHY REGIONS
LEVEL
  STORE
  ,SALES_REGION
```

HIERARCHY (Dimension)

HIERARCHY STATES

LEVEL

STORE

,STATE

,GEO_REGION

6.82 IF-OTHERWISE

Use

IF-OTHERWISE assigns a value to a variable if a condition is either true or false.

Syntax

```
<variable>
  IF <condition>
    = <expression>
  [IF <condition>
    = <expression>
    .
  ]
  [OTHERWISE
    = <expression>]
```

Parameter	Description
<variable>	Name of a variable. Variable names that use special characters should be in single quotation marks (' ').
IF	Indicates a conditional block. If the condition is true, assigns the value of the expression to the variable. If the condition is false, proceeds to the next conditional block (if it exists) or to the OTHERWISE block (if it exists). If none of the conditions is true and there is no OTHERWISE block, the value of the variable is unchanged.
<condition>	A logical condition using any arithmetic or logical operators.
<expression>	An arithmetic expression.
OTHERWISE	Indicates a conditional block. If all previous conditions are false, assigns the value of the expression to the variable.

Example

...This example assigns values to the variable **Commission**, depending on the value of **Sales**:

Commission

```
IF Sales GT 1000 = .2
IF Sales GT 500 = .15
IF Sales GT 250 = .125
OTHERWISE = .1
```

6.83 IN (OF) FROM VERSION

Use

IN (OF) FROM and VERSION are qualifiers that reference specific variables.

Syntax

```
<variable> {IN|OF} <dimension> <member> [...{IN|OF}<dimension> <member>]
  <variable> FROM <database>
  <variable> VERSION <version>
```

Note: IN and OF are equivalent and, therefore, interchangeable.

Parameter	Description
<variable>	Name of a variable. Variable names that use special characters should be in single quotation marks (' ').

INDEX-CASE-ENDINDEX

<dimension>	Name of a dimension.
<member>	Short name of a member in the dimension. Dimension member names that use special characters should be in single quotation marks (' '). Note: Because you can have duplicate labels (long names), you must use the short member name when using the IN construct.
<database>	Name of an attached database.
<version>	Name of an existing version.

Examples

...You can use commands like the following in logic sets:

Sales OF Product Cars IN Region Buffalo FROM corpdb VERSION optimistic

...In some financial models you need to allocate a cost at a consolidated level of an organization to ... the input levels of the organization. For example, you can report overhead costs in a corporation ... at the corporate level, but allocate them to the lowest levels of the organization on the basis of ... their revenue as a percentage of total corporate revenue. IN and OF allow you to allocate costs in ... this way. Assume you have the following Company dimension:

INPUT east, west, north, south

OUTPUT northwest, southeast

RESULT corporate

northwest = SUM north, west

southeast = SUM south, east

corporate = SUM northwest, southeast

...You also have the variables Revenue and Overhead dimensioned by Company, and an ... undimensioned variable Corpoverhead. To allocate the corporate overhead, you first enter data ... values for Corpoverhead. Then enter the revenue in the input sections of the dimension. Next use ... CONSOLIDATE to consolidate the revenue. Finally, in a logic, you might have this expression:

Overhead = (Revenue / Revenue IN Company Corporate)*Corpoverhead

...When you calculate the logic, Application Server takes the revenue in each member of the ... dimension, and divides that figure by the consolidated revenue in the corporate member. ... Application Server then multiplies that figure by Corpoverhead. This calculated overhead figure is ... then put in the variable Overhead as the current member.

6.84 INDEX-CASE-ENDINDEX

Use

INDEX-CASE-ENDINDEX executes logic statements for the specified dimension members.

Syntax

```
INDEX <dimensions>
  CASE <members>
    .
    . <statements>
    .
    [CASE members
      .
      . <statements>
```

```
.]
ENDINDEX
```

Parameter	Description
INDEX	Indicates the start of an INDEX block. You can specify the dimensions to use when evaluating a CASE statement.
<dimensions>	One or more dimension names separated by commas.
CASE	Executes the block of statements if the specified members are members of the dimensions in the INDEX statement.
<members>	Name of one or more members, classes, or levels separated by commas. You can use an asterisk (*) to indicate any member name (default).
<statements>	A sequence of statements or commands.
ENDINDEX	Indicates the end of an INDEX block.

Example

...In this example, two constants, Erg Ratio and Msr Ratio, are set to

...specific values depending on the values of their dimensions, which are Product and Company:

INDEX Company, Product

CASE GM, Cars

 'Erg Ratio' = 3.61

 'Msr Ratio' = 0.85

CASE Ford, Trucks

 'Erg Ratio' = 2.55

 'Msr Ratio' = 0.98

CASE ,

 'Erg Ratio' = 3.02

 'Msr Ratio' = 0.88

ENDINDEX

...INDEX-CASE-ENDINDEX sets Erg Ratio and Msr Ratio to specific values

...for the combinations GM, Cars and Ford, Trucks. Otherwise, their values are 3.02 and 0.88.

6.85 INDEX USER-CASE-ENDINDEX

Use

INDEX USER-CASE-ENDINDEX executes SELECT statements for individual users or groups of users and thereby restricts access to dimensions. You use the INDEX USER construct in a procedure called Security, which Application Server executes each time that you use or attach the database.

Note: Only the user or group with the same name as the database can create and edit the Security procedure. Use the CREATE USER or CREATE GROUP command to create such a user or group if one does not yet exist.

You create a separate Security procedure in each database where you want to implement security features.

Syntax

INDEX USER

```
    CASE {<user> | <group> | *}
```

INDEX USER-CASE-ENDINDEX

SELECT <dimension> [NONE]

ENDINDEX

Parameter	Description
INDEX USER	Indicates the start of an INDEX USER block.
CASE	Restricts user or group access based on the specified SELECT command(s). You can specify multiple cases, up to a maximum of 1,000 cases, within the INDEX USER-ENDINDEX construct.
<user>	Name of a user defined in MASTERDB whose dimension view you want to restrict.
<group>	Name of a group defined in MASTERDB whose view Application Server will specialize for all the users in the group.
*	Specifies that all users not previously mentioned in a CASE statement will follow the restrictions set by the specified SELECT commands.
SELECT <dimension>	One or more valid Application Server SELECT commands specifying a view of the data for the user or group. Omitting a SELECT <dimension> command for a CASE implies that the whole dimension is selected and available for the users in the CASE.
NONE	Prohibits the user or group from viewing the whole dimension. Tip: After you define all the cases for users and groups, you can globally prohibit all other users from all areas of the database using the following construct: CASE * SELECT <dimension> NONE SELECT <dimension> NONE ...

Rules

The Security procedure executes each time that you use or attach the database.

If a user belongs to multiple groups, Application Server uses the first CASE statement that matches the user's name or any group the user is a member of to provide the restricted view.

A user or group with the same name as the Use database is the Security procedure administrator. That user or all users in the group can edit the Security procedure. For example, if you are adding a Security procedure to the DEMO database, any user who is a member of a group named Demo or any user named Demo can edit the Security procedure.

The following users have an unrestricted view of all dimensions:

Any user with the same name as the database.

Any group of users with the same name as the database.

Any user you have omitted from all groups and from all CASE statements in a Security procedure.

If you specify a CASE statement that executes a SELECT <dimension> NONE for each dimension, Application Server restricts the user or group specified in the CASE from all dimensions. If you specify CASE *, Application Server restricts all users whom you have not assigned to a group from all dimensions.

Example

...This example defines the Security procedure for the Demo database:

INDEX USER

CASE Eastmgr

... Specifies the view for the Eastmgr group

```
SELECT Region Below East
SELECT Type Budget
SELECT Product Outputs
CASE DEMO
... DEMO group users are administrators; they can edit the procedure.
... All members of group DEMO get full dimension access.
... Nothing is selected at the time a database is attached or used.
CASE FINANCE
... Specifies the view for the Finance group
SELECT Region South
SELECT Product Inputs
SELECT Type Actual
CASE *
... Restricts all dimension views for all other users.
SELECT Product NONE
SELECT Region NONE
SELECT Type NONE
ENDINDEX
... This example shows how to restrict the view of an attribute. It defines
... which Customers the users in the West group can see, and as a result,
... which members of the attribute State:
INDEX USER
CASE West
... Specifies the view for the West group
SELECT Customer where State is CA
SELECT Customer plus where State is OR
SELECT Customer plus where State is WA
ENDINDEX
```

6.86 INPUT (Dimension)

Use

INPUT is a Dimension keyword that identifies the members for the lowest level of data in a database.

Note: The members that you define in an INPUT statement are called *input members*. Member names and labels are sometimes referred to as short names and long names. When displaying data, use SET SHORT|LONG to display the member names or labels.

Syntax

```
INPUT
.
. <member> [ '<label>' ]
.
```

INT()

Parameter	Description
INPUT	Specifies the input members for a Dimension set.
<member>	Name of the input member, up to 96 bytes (including periods (.) and underscores (_)). The parameter <member> must begin with a letter. Dimension member names that use special characters should be in single quotation marks (' '). The INPUT section can contain multiple members. Note: The parameter <member> cannot start with an ampersand (&) followed by numbers. For example, the following statement cannot be used: INPUT '&123' 'invalid member'
'<label>'	A text string to assign to the input member, up to 250 characters, enclosed in single quotation marks (' '). A label is a long description of the input member name. A recommended label length is no more than 30 characters. Note: You cannot specify a label that begins with an asterisk and then a blank space. You can use other characters though. For example, you cannot specify '* Massachusetts' but you can specify the label '~ Massachusetts'.

INPUT (Dimension) - Rules

You should not use member names that are valid Application Server keywords. For example, a member name called NE might be confused with a NE (not equal) operator during a SELECT. If you must use an Application Server keyword, enclose the name in double quotation marks (" ").

You must use unique member names.

Member labels do not need to be unique. If member labels are not unique, it may be confusing to end users because they will not be sure which members they are looking at.

An INPUT statement appears after any ALLOCATE or KEY BOTH statements and before any OUTPUT, HIERARCHY, RESULT, or consolidation statements.

Usually, you specify one INPUT statement, followed by a list of member names and, optionally, labels. Except for the last line, you must separate <name> '<label>' statements with a comma. You can specify more than one INPUT statement if you want to group input members together; the functionality of the Dimension set does not change.

Example: INPUT

```
INPUT
'80286' '286 Chip',
'80287' '287 Chip',
'80386' '386 Chip'
OUTPUT ...
```

6.87 INT()

Use

INT rounds to the nearest integer.

Syntax

CALCULATE <result> = INT(<variable>)

Parameter	Description
<result>	Name of the result variable.

<variable> Name of the variable whose data you want rounded to the nearest integer.
Variable names that use special characters should be in single quotation marks (' ').

Example

...This example returns a value of -4:

b = -4.1

CALCULATE a = INT(b)

6.88 IRR()

Use

IRR returns a Scalar value for the internal rate of return. Application Server calculates the rate relative to the first payment.

Syntax

CALCULATE <returnrate> = IRR(<amt1>, <date1>, <amt2>, <date2>, ..., <amtn>, <daten>)

Parameter	Description
<returnrate>	Name of the result variable with the scalar internal rate of return.
<amtn>	Value of the <n> th payment, where <n> = 1 to total payments.
<daten>	Date of the <n> th payment, where <n> = 1 to total payments.

6.89 IS

Use

IS references the period of the current observation.

Syntax

DAY IS <day>
MONTH IS <month>
QUARTER IS <quarter>
<variable> IS {RATE | EXPENSE}
YEAR IS <year>

Parameter	Description
DAY IS	Indicates the period is a day.
<day>	Name of a day: Sunday, Monday, ..., Saturday.
MONTH IS	Indicates the period is a month.
<month>	Name of a month: January, February, ..., December.
QUARTER IS	Indicates the period is a quarter.
<quarter>	Name of a quarter: first, second, third, or fourth.
<variable>	Name of the variable being consolidated (used only in dimensions). Variable names that use special characters should be in single quotation marks (' '). Can be used with the following: RATE - Declares the variable as a rate variable with SET VARIABLE. EXPENSE - Declares the variable as an expense variable with SET VARIABLE.
YEAR IS	Indicates the period is a year.
<year>	A numeric year format, for example, 1999.

JOB

Example

...This example sets the variable Discount to .10 in 1999 and to .07 in other years:

WHEN year is 1999

Discount = .10

ELSE

Discount = .07

ENDWHEN

...This expression (in a dimension) reverses the sign of the Variance for Expense variables:

WHEN variable IS Expense

Variance = Budget - Actual

ELSE

Variance = Actual - Budget

ENDWHEN

6.90 JOB

Use

JOB executes the commands in a procedure or in an external file, and displays them on your screen when Application Server executes them.

Syntax

JOB <procedure> <setname>[:<database>] | <filename>[:EXTERNAL] [RESOLVE]

Parameter	Description
<setname>	Name of a set, such as a logic set or a dimension set.
<procedure>	Name of a procedure. If you do not specify a name, Application Server uses the default procedure (if defined), or the last procedure you edited.
<filename>	Name of an external file that the SUPERVISOR DUMP command creates that contains user and database information about MASTERDB.
<database>	Name of the database where the procedure is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the procedure is a text file that is not in an Application Server database.
RESOLVE	If you specify this parameter, Application Server recompiles the procedure before executing it. This ensures that Application Server uses the current value of control variables in the set.

Remarks

If you omit the ;EXTERNAL keyword and the ;<database> parameter, Application Server searches for a set named <setname> within the Use database. If a set named <setname> is not found, Application Server searches for an external file.

Example

...This example executes the MERCHSEL procedure, which selects a

...dimension and variable and defines a time period and the across/down dimensions:

JOB merchsel

...Application Server writes the following commands to the output stream:

SELECT Merchandise

SELECT Sales

SET monthly period 90

ACROSS Variables, Time DOWN Merchandise

...This example execute the DOS/Windows file c:\myjob:

JOB c:\myjob

6.91 KEY (Dimension)

Use

KEY is a Dimension command and a CONSTRUCT keyword that creates an index based on the dimension member names and labels. If you key a dimension you can quickly select its members when you select the members by member name and label.

Note: KEY is not necessary for small dimensions with fewer than 50 members.

Syntax

KEY [BOTH]

Parameter	Description
BOTH	Creates an index based on both dimension member names and labels.

Rules

You can add the KEY BOTH keyword to the first line of the Dimension editor for the specified dimension. You can also add KEY BOTH using the PREFACE keyword in the CONSTRUCT command.

You can remove the indexes created during a KEY BOTH by removing the keyword from the Dimension editor and then recompiling the dimension.

Example

...This example shows how KEY BOTH is added to the first line in the Dimension editor. When

... Application Server compiles the dimension, it will be keyed. Selecting dimension member names

... or labels is optimized because the dimension is keyed.

KEY BOTH

ALLOCATE 100, 10, 1

INPUT ACME_MFG 'Acme Manufacturing'

,AZTEC_MACH_TOOL 'Aztec Machine Tools'...

6.92 KILL (Supervisor)

Use

KILL is a Supervisor command that terminates another client/server session.

When Application Server is running on Windows, the KILL command forces the program to exit gracefully. No orphan Work databases remain. When Application Server is running in Linux/UNIX, the KILL command forces the program to abruptly exit, and any associated Work databases may remain as an orphan and may not be cleared or removed. To clear and remove orphaned Work databases, use the Supervisor CLEAN WORK DATABASES command.

LAG()

Syntax

KILL [CONNECTED [TO] <database>] [USER] <username>[(<session>)] | [PROCESS] <processid>

Parameter	Description
[CONNECTED [TO] <database>]	Terminates all users connected to the specified <database>.
[USER] <username>	The name of the user session to terminate.
<session>	The session number.
[PROCESS] <processid>	The operating system process number to terminate.
Note: You cannot terminate a non-client/server session.	

Examples

...This example terminates the user ADMIN:

KILL USER ADMIN

...This example terminates the operating system process 13567:

KILL 13567

6.93 LAG()

Use

LAG moves data to the right by the specified number of time periods.

Syntax

CALCULATE <result> = <variable> LAG <number>

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable containing data you want to move. Variable names that use special characters should be in single quotation marks (' '). Note: Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.
<number>	Number of time periods to move data to the right.

Example

...EXAMPLE 1: Consider the following statement:

CALCULATE 'Tax Paid' = 'Tax Due' LAG 1

...The results in a report that prints Tax Due and Tax Paid look like:

Tax Due	135	142	168	129
Tax Paid	----	135	142	168

...EXAMPLE 2: This separate example assumes the following data for a variable called Myvar:

Myvar	200	300	400	500
-------	-----	-----	-----	-----

...The following command moves Myvar data one column to the right:

CALCULATE 'Myvar' = 'Myvar' LAG 1

...Myvar data now appears as follows:

Myvar	----	200	300	400
-------	------	-----	-----	-----

6.94 LAST()

Use

LAST returns a value from a previous period.

Syntax

CALCULATE <result> = <variable> LAST <period>

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable containing data you want to retrieve. Variable names that use special characters should be in single quotation marks (' ').
<period>	The period for which you want a value: year (default), quarter, month, or week.

Example

...This example defines Growth as the value of Sales minus the value of Sales Last Year:

CALCULATE Growth = Sales - Sales Last Year

6.95 LATEST

Use

LATEST returns the position of the latest month, week, day, and so on, in the fiscal year.

Syntax

LATEST <periodicity>

Parameter	Description
<periodicity>	The period you want returned: quarter, month, week, day, and so on. If you use MONTH, Application Server returns the values 1-12.

6.96 LEAD()

Use

LEAD shifts data to the left by the specified number of periods.

Syntax

CALCULATE <result> = <variable> LEAD <number>

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable containing data you want to shift. Variable names that use special characters should be in single quotation marks (' '). Note: Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.
<number>	Number of time periods to shift data to the left.

Example: LEAD

...Consider the following statement:

CALCULATE 'Tax Paid' = 'Tax Due' LEAD 1

LEVEL (Dimension)

...The results in a report that prints Tax Due and Tax Paid looks like this:

Tax Due	----	135	142	168
Tax Paid		135	142	168
			129	

6.97 LEVEL (Dimension)

Use

LEVEL is a Dimension command that defines the members and the level within a dimension. You can select that group of members by specifying the level name instead of selecting the member names individually.

Syntax

LEVEL <level>, <level>, ..., <level>

Parameter	Description
<level>	Name of a level up to 24 alphanumeric characters. The first level always refers to input members. The second level refers to the members that the first level rolls into. The third level refers to the members that the second level rolls into, and so on. A dimension can have up to 20 levels.

Note: Referencing a level in a logic is meaningful only if that level is above the current section in the consolidation. As shown in the example, it would be meaningless to reference a zone when performing a calculation at the state level, because the zone could have numerous values for a given state.

Rules

Level statements must appear after input, output, and result statements, and before any consolidation statements.

Example

...In this example, the dimension Region contains as its input members the distributors for a ... company's product line. The distributors are grouped into zones. Zones are grouped into ...states. States form the total region.

INPUT d4010, d4014, d4019, d5110, d5214,

OUTPUT z40, z50, z51, z52,

OUTPUT California, Texas, Washington,

RESULT Total_Region

LEVELS Distributor, Zone, State

z40 = SUM d4010, d4014

z51 = d5110

.

California = SUM z40, z51

Texas = SUM z52, z53

.

Total_Region = SUM California, Texas, Washington,

...In a logic set you can use all the commands that refer to dimension

...members using the names Distributor, Zone, and State. For example:

Advertising = Advertising IN Region State * Sales/Sales IN Region State

...This statement allocates state advertising expenses to a given distributor based on the ratio of that distributor's sales to the total sales of all the distributors in the state. Similarly:

SELECT Region State

...selects those members at the level of State, namely California, Texas,

...Washington, and so on.

6.98 LIST

Use

LIST displays a default report, according to the current across/down selections.

Syntax

```
LIST [COLUMNS <n>] [STACK] [FULL] [PASTE] [PAGE ON <dimensions>]
      [TITLE '<title>'] [CTITLE <column>, '<title>' [,..., <column>, '<title>']] [ <periodicity> ] [PERIOD
      <daterange>]
      [TABS] [WIDTH] [LENGTH]
```

Parameter	Description
COLUMNS <n>	Lists the specified number of columns across the page. If more columns of data exist, they wrap to the next section.
STACK	Stacks column headings over each column.
FULL	Prints all rows, even if all values in a row are missing. By default, Application Server skips missing rows.
PASTE	Prints all values in a row when there are more values than there are columns and some of the data is missing. If the missing values fit on a physical row, you can use PASTE to show the missing values. For example, if you had 12 months of data for the variable sales, and your monitor displayed 6 columns across, Application Server would display the data as 2 rows of six columns each. If the values were missing for the first six months, PASTE would show six months of missing values, rather than skipping the first row of data entirely.
PAGE ON <dimensions>	Forces page breaks whenever the next member of the specified dimension(s) prints. Restricted to dimensions specified in the DOWN command.
TITLE '<title>'	Prints a title at the top of the listing.
CTITLE <column>, '<title>'	Specifies the column number and the title that will appear above that column. You can specify one or more column numbers and titles, each separated by a comma. For example: LIST monthly CTITLE 1, 'Standard', 2, 'Advanced'
<periodicity>	Displays the data at one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, or hourly.
PERIOD <daterange>	Limits the display to a specified date range, where <daterange> is a range of dates separated by a dash (-). For example, 00/12/3 - 00/12/31 is the period from December 3, 2000 through December 31, 2000, and 99 - 00 is all of 1990 and 2000. When your variables have been selected and the report layout is determined, you can enter the LIST command with the PERIOD keyword or LIST by itself. The results appear in the display window. The LIST command generates a simple report listing based on the selected data, layout, and defined time frame. It is a very efficient way to display the data because all the defaults are used. If no period keyword is entered, the entire time frame is displayed. Limiting the time frame might make it easier to view the data.

LN()

TABS <i>character</i>	Application Server displays the fields in the LIST output separated by tab characters, or by the optional literal character specified after the TABS keyword. You must enclose the symbol in single (' ') or double (" ") quotation marks. Use TAB 'symbol' when the report set uses the TABS keyword but you want to use a different symbol so that you can easily transfer the report into Microsoft Excel.
WIDTH <width>	Sets the maximum number of characters that can appear on a line.
LENGTH <length>	Sets the maximum number of lines that can appear on a page.

Example

...This example displays the selected variables with a monthly periodicity:

LIST monthly

...This example displays the selected variables with a weekly periodicity

...for the period from June 1 to September 30, 1999:

LIST weekly PERIOD 99/6 - 99/8/30

...In this example Application Server displays the fields separated by an asterisk:

LIST TABS '**'

6.99 LN()

Use

LN returns the natural logarithm (base e) of a value or variable.

Syntax

CALCULATE <result> = LN(<variable>)

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable or arithmetic expression whose natural logarithm you want to calculate. Variable names that use special characters should be in single quotation marks (' ').

6.100 LOAD

Use

LOAD loads a database from an external dump file of any size.

Syntax

LOAD <file>

Parameter	Description
<file>	Name of an external text file, such as a file containing a database created with DUMP or SNAPSHOT. If you specified a wildcard when you dumped the database, for example DUMP file??, you will have multiple dump files. Therefore, you must use the same wildcard to load the database, for example LOAD file??.

Rules

DUMP creates a file that first erases any set or variable it contains. If you load a file into a database that is not empty, the sets or variables being loaded erase existing sets or variables with the same name.

Example

...On one computer, you can dump a database to an external file named

...MARKETDB using the following statement:

```
DUMP marketdb
```

...After transferring the file to another machine, you can load it into a

...newly created database using the following statement:

```
LOAD marketdb
```

...This example loads the contents of the tape named "/dev/rmt/0u":

```
LOAD TAPE /dev/rmt/0u
```

6.101 LOAD (Supervisor)

Use

LOAD is a Supervisor command that loads MASTERDB from an external dump file or tape.

Syntax

```
LOAD <setname>[:<database>] | <filename>[:EXTERNAL]
```

Parameter	Description
<setname>	Name of set.
<filename>	Name of an external file that was created by the SUPERVISOR DUMP command that contains user and database information about MASTERDB.

Note: If you are trying to load a dump of MASTERDB that was dumped prior to version 9.3 of Application Server and contains information about partitioned Application Server databases, the dump file will be unloadable. You must edit the dump file to remove the PARTITION and NAME clauses from the ADD DATABASE commands, and then you will be able to load the pre-9.3 version of MASTERDB. For help doing this contact Customer Support.

Example

...This example loads the contents of the file MASTER:

```
LOAD master
```

6.102 LOG()

Use

LOG returns the base 10 logarithm of a value or variable.

Syntax

```
CALCULATE <result> = LOG(<input>)
```

Parameter	Description
<result>	Name of the result variable.
<input>	Name of the variable or arithmetic expression whose base 10 logarithm you want to calculate.

Example

...This example returns a value of 2.5623:

```
b = 365
```

LOGIC

CALCULATE a = LOG(b)

6.103 LOGIC

Use

LOGIC starts the Application Server editor, where you can create or edit logic sets.

Syntax

LOGIC [<logic>] [;<database> | ;EXTERNAL|LOCAL]

Parameter	Description
<logic>	Name of a new or existing logic set up to 96 bytes. If you do not specify a name, Application Server uses the default logic set if you defined it, or the last logic set you edited.
<database>	Name of the database where the logic set is located. If you do not specify a database name, Application Server uses your Use database.
EXTERNAL	Indicates the logic set is a text file that is not in an Application Server database. If the logic set is a DOS file, its name cannot have an extension.
LOCAL	Indicates that the logic set is a text file on the client.

Note: You can create or edit a logic with the EXTERNAL or LOCAL extension, but you must copy it into a model and compile it before it can be used in the model.

Rules

SAVE and END perform a syntax check on external logic sets and then copy the new or modified logic sets back to the external file.

Application Server executes a logic set when you enter the CALCULATE command.

Example

...This example creates a logic set REVENUE:

LOGIC revenue

Revenue = 'Unit Price' * Volume

Discounts = Revenue * ('Discount Rate' / 100)

'Net Revenue' = Revenue - Discounts

'Variable Cost' = Volume * 'Unit Var Cost'

6.104 LOOP-ENDLOOP

Use

LOOP-ENDLOOP repeatedly executes a block of statements until a condition becomes true.

Syntax

```
LOOP
    .
    . <statements>
    .
    EXITIF <condition>
    .
    . <statements>
    .
ENDLOOP
```

Parameter	Description
LOOP	Indicates the start of a loop and executes the first block of statements until an EXITIF condition is true.
<statements>	Any valid sequence of statements or commands.
EXITIF	Executes the next block of statements when the condition is true.
<condition>	A logical condition using any arithmetic or logical operators.
ENDLOOP	Indicates the end of a loop.

6.105 MASK-ENDMASK

Use

MASK-ENDMASK is a Report command that creates a template for the report rows. All nonblank characters in the mask replace whatever normally appears in the rows that follow until an ENDMASK occurs.

Syntax

```
MASK [<n>] '<text>', ... [<n>] '<text>'
      .
      . <statements>
      .
      ENDMASK
```

Parameter	Description
<n>	A positive integer indicating the number of spaces to indent text from the left margin.
<text>	Text or characters you want to display.
<statements>	Any sequence of statements or commands.

Note: Make sure that you do not mask more columns than the number of columns of data. MASK-ENDMASK is independent of the number of columns displayed and formats columns even if no data is displayed.

Example

...The following block inserts vertical bars between the numbers in a report:

```
MASK ' | | | '
ROW Sales
ENDMASK
```

...This block produces output similar to:

```
Sales 459 | 512 | 505 |
```

6.106 MAX()

Use

MAX returns the maximum value of the specified variables.

Syntax

```
CALCULATE <result> = MAX(<variable> [...], <variable>))
```

Parameter	Description
<result>	Name of the result variable.

MEAN()

<variable>

Names of the variables to search for the maximum value. Separate each variable with a comma. If you specify only one variable, Application Server returns the maximum value of the entire time series for the variable, and uses that value in all the time periods for the result variable. If you specify more than one variable, Application Server returns the maximum value of variables in each time period and uses that value in the corresponding time period for the result variable.

Variable names that use special characters should be in single quotation marks (' ').

Example

...Consider the following statement:

CALCULATE Maxcost = MAX(Costa)

...The results in a report that prints Costa and Maxcost would be:

Costa 10 3 9 1

Maxcost 10 10 10 10

6.107 MEAN()**Use**

MEAN returns the average of the specified variable, where any missing values are excluded and not counted in the divisor.

Note: MEAN behaves like MAX in that if you specify one variable, Application Server returns the mean value of the entire time series for the variable, and uses that value in all the time periods for the result variable. If you specify more than one variable, Application Server returns the mean value of variables in each time period and uses that value in the corresponding time period for the result variable.

Syntax

CALCULATE <result> = MEAN(<variable> [<variable>,<variable>,... <variable>])

Parameter	Description
<result>	Name of the result variable.
<variable>, ...	Name of the variables whose average value you want.

6.108 MEAN2()**Use**

MEAN2 returns the average of the specified variable, and allows you to specify whether to include or exclude missing observations in the average, or to return a missing value if at least one observation is missing.

Syntax

CALCULATE <result> = MEAN2(<type>, <variable> [<variable>,... <variable>])

Parameter	Description
<result>	Name of the result variable.
<type>	An integer value 0, 1, or 2. 0 means if any of the observations are missing, mark all observations as missing. 1 specifies an exclusive average, where missing values are not counted in the divisor. 2 specifies an inclusive average, where missing values are counted in the divisor.
<variable>,...	Name of the variables whose average value you want.

6.109 MIN()

Use

MIN returns the minimum value of the specified variables.

Syntax

CALCULATE <result> = MIN(<variable> [,..., <variable>])

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variables to search for the minimum value. Separate each <variable> with a comma. Variable names that use special characters should be in single quotation marks (' ').

Example

...This example returns values of 1, 6, -8, 72, 3:

b = 1, 10, 4, 72, 3

c = 100, 6, -8, 102, 19

CALCULATE a = MIN(b, c)

6.110 MORTGAGE()

Use

MORTGAGE returns four values: the series of principal payments, the series of interest payments, the total of interest and principal payments, and the remaining balance.

Note: MORTGAGE is a user-defined function that calculates more than one output variable. Application Server writes it as a logic set and stores it in the database TBDB. You can enter the following statement to look at the database:

type mortgage;tddb

Syntax

MORTGAGE (<amount>, <annual_rate>, <term>, <start>, <first_pay>, <balloon>, <principal>=, <interest>=, <payments>=, <balance>=)

Parameter	Description
<amount>	Scalar amount of the mortgage.
<annual_rate>	Scalar annual interest rate.
<term>	Scalar length of the mortgage in years.
<start>	Starting date of the mortgage.
<first_pay>	Date of the first payment.
<balloon>	Scalar amount of any balloon payment, or 0 if there is no balloon payment.
<principal>	Name of the output variable for the series of principal payments.
<interest>	Name of the output variable for the series of interest payments.
<payments>	Name of the output variable for the series with the total of interest and principal payments.
<balance>	Name of the output variable for the series of remaining balances on the mortgage.

MOVING()

6.111 MOVING()

Use

MOVING returns a moving average for a previously defined data range. MOVING returns a missing value if any of the observations are missing.

Syntax

CALCULATE <result> = MOVING(<variable>, <number>)

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable whose moving average you want. Variable names that use special characters should be in single quotation marks (' '). Note: Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.
<number>	Number of time periods to average.

6.112 MOVING2()

Use

MOVING2 returns a moving average for a previously defined data range and allows you to specify whether to include or exclude missing observations in the average, or to return a missing value if at least one observation is missing.

Syntax

CALCULATE <result> = MOVING2(<variable>, <type> [, <number>])

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable whose moving average you want. Variable names that use special characters should be in single quotation marks (' '). Note: Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.
<type>	An integer value 0, 1, or 2. 0 means if any of the observations are missing, mark all observations as missing. 1 specifies an exclusive average, where missing values are not counted in the divisor. 2 specifies an inclusive average, where missing values are counted in the divisor.
<number>	Number of time periods to average for every periodicity. If you do not specify a <number>, the number of periods over which the moving average is calculated is taken as a predefined default value that is different for each periodicity. For example, if your periodicity is MONTHLY the default value is 12. The default list of period numbers is as follows: BIENNIAL DefaultValue=2 YEARLY DefaultValue=2 RYEARLY DefaultValue=2 YTD DefaultValue=2 AVGYTD DefaultValue=2 SEMIANNUAL DefaultValue=2 QUARTERLY DefaultValue=4 RQUARTERLY: DefaultValue=4

QTD DefaultValue=4
 AVGQTD DefaultValue=4
 BIMONTHLY DefaultValue=6
 MONTHLY DefaultValue=12
 RMONTHLY DefaultValue=12
 MTD DefaultValue=12
 AVGMTD DefaultValue=12
 MYTD DefaultValue=12
 LUNAR DefaultValue=13
 LTD DefaultValue=13
 PBIWEEKLY DefaultValue=2
 BIWEEKLY DefaultValue=2
 WKTD DefaultValue=4
 WEEKLY DefaultValue=4
 BSDAILY DefaultValue=7
 DAILY DefaultValue=7
 HOURLY DefaultValue=24
 QHOURLY: DefaultValue=4

Overriding the defaults

By defining control variables, you can customize the number of periods averaged for each periodicity. When you omit the <number>, the MOVING2 function will use the number of periods defined by the control variables rather than use the defaults for the periodicity. The control variables are in the form of:

MOVING_ <periodicity>

where <periodicity> matches the names shown in the default list. For example, if your periodicity is MONTHLY, and you issue the following command:

SET CONTROL MOVING_MONTHLY 4

it will average 4 months of data when measures are listed monthly instead of the default 12 periods

The SET CONTROL commands should be added to an AUTOUSE procedure so they are executed every time the model is used.

Example

...This example calculates a moving average for the first four periods (Jan 00 - Apr 00) of the variable

... Cogs and puts the average in the Apr 00 entry of the variable x. The May 00 entry

...of x contains the average of Cogs for Feb 00 to May 00.

CALC x = MOVING(cogs, 4)

6.113 NCASES()

Use

NCASES returns the maximum number of observations (based on the periodicity) for the period of the logic set being calculated.

Syntax

CALCULATE <result> = NCASES()

Parameter	Description
<result>	Name of the result variable.

NDAYS()

6.114 NDAYS()

Use

NDAYS returns the number of days in the current period.

Syntax

CALCULATE <result> = NDAYS()

Parameter	Description
<result>	Name of the result variable.

6.115 NEWPAGE

Use

NEWPAGE is a Report command that starts a new report page.

Syntax

NEWPAGE

6.116 NOCALC-ENDNOCALC

Use

NOCALC-ENDNOCALC is a Report command that defines a group of rows that should not have calculated columns.

Syntax

```
NOCALC
.
. <row specifications>
.
ENDNOCALC
```

Example

...Assume the columns in your report are in the following order:

```
PRODUCTS TV
  Apr 00    May 00
Sales  15,355 15,721
Expenses    7,439 7,859
```

...If the report is:

```
SET
ORDER 0, 1, 2, 'Pct' = c2 % c1
ENDSET
ROW Sales
NOCALC
  ROW Expenses
ENDNOCALC
```

...The report would look like this:

PRODUCTS TV

	Apr 00	May 00	Pct	
Sales	15,721	15,355	98	
Expenses		7,859	7,439	---

6.117 NPV()

Use

NPV returns a scalar net present value of a series of payments discounted at a given rate.

Syntax

CALCULATE <presentvalue> = NPV(<rate>, <base_date>, <amt1>, <date1>, ..., <amtn>, <daten>)

Parameter	Description
<presentvalue>	Name of the result variable that contains the net present value as a scalar value.
<rate>	Name of the scalar or time-series variable containing the discount rate.
<base_date>	Date that Application Server bases its calculations on.
<amtn>	Value of the <n> <i>th</i> payment, where <n> = 1 to total payments.
<daten>	Date of the <n> <i>th</i> payment, where <n> = 1 to total payments.

6.118 NROWS()

Use

NROWS is a Report function that returns the number of rows in the last down dimension used for a report. Note that you can only use this function in reports, not in normal CALCULATE commands.

Syntax

CALCULATE <result> = NROWS()

Parameter	Description
<result>	Name of the result variable.

Example

...This example displays all the rows in the last down dimension, with a blank line between each row:

scalar i

DO i = 1, NROWS()

 ROW [i]

 SKIP

ENDDO

6.119 OBS()

Use

OBS returns the number of the individual observation being calculated at any given step of a logic set. For example, if the periodicity of a logic set is weekly, and the time period is for one year, OBS returns a value of 1 when the first week is calculated, a value of 2 when the second week is calculated, and so on, up to a value of 52 when the last week is calculated.

Note: The maximum number of observations is equal to the value returned by NCASES.

ON

Syntax

CALCULATE <result> = OBS()

Parameter	Description
<result>	Name of the result variable.

6.120 ON

Use

ON executes a specified procedure when either an error or a cancel condition occurs.

Syntax

ON {ERROR | BREAK} <action> | CLEAR

Parameter	Description
ERROR	Executes the specified <i>action</i> if an error occurs.
BREAK	Executes the specified <i>action</i> if a cancel condition occurs.
<action>	Application Server command or the name of a procedure to be executed if an error or cancel condition occurs. The parameter must be enclosed in quotes.
CLEAR	Stops execution of the specified <i>action</i> when an error or break key condition occurs.

Note: You should test an error procedure before including it in an ON ERROR command to make sure it does not contain any errors. An error procedure that contains errors could result in an endless loop.

6.121 ORDER

Use

ORDER changes the order in which Application Server displays dimension members.

Syntax

```
ORDER <dimension>
  [ALPHABETIC | LABELS]
  [ASCENDING]
  [ HIERARCHY [ALLHIERARCHIES] [ ALPHABETIC | LABELS ] ]
  [ON [ASCENDING] <expression>]
  [TOP / BOTTOM {<n> | NSELECTED | ALLMEMBERS }
    [CONCENTRATE | REMAINDER
      [CONCENTRATE <options> ] ] [<member criteria>] ]
  [<members>]
  [SELECT]
  [ACROSSDOWN]
```

Parameter	Description
<dimension>	Name of the dimension to order. If you do not specify options after the dimension, Application Server restores the dimension to its default order.
ALPHABETIC	Orders members alphabetically.
LABELS	Orders members alphabetically based on their labels.
ASCENDING	Displays members in ascending order: input members first, followed by output members, and then result members.
HIERARCHY	(This applies to dimensions with multiple hierarchies) Orders members by the default hierarchy set for the dimension.

	Note: You can set a hierarchy using the SET command. If you do not set a hierarchy, Application Server uses the default hierarchy (the one defined in the first HIERARCHY statement of the CONSTRUCT command).
ALLHIERARCHIES	For multiple hierarchies, where members are present in more than one hierarchy, displays all members of all hierarchies wherever they occur.
ALPHABETIC	Orders members of the default hierarchy alphabetically.
LABELS	Orders members of the default hierarchy alphabetically based on their labels.
ASCENDING	Displays members in ascending order: input members first, followed by output members, and then result members.
ON	Orders members from the member with the largest numeric value to the member with the smallest numeric value according to the specified expression.
ASCENDING	Orders members from the member with the smallest numeric value to the member with the largest numeric value according to the specified expression.
<expression>	An arithmetic expression.
TOP BOTTOM	Orders the top or bottom selected members of the dimension, based on a number, or the selected members, or all members. Any output from a LIST command will list the specified items in top or bottom order.
<n>	Orders just the top or bottom <n> selected members of the dimension. Any output from a LIST command will list n items in top or bottom order.
NSELECTED	Determines n from the number of currently selected members. For example, if ten members of the Product dimension are currently selected, then: ORDER PRODUCT ON SALES TOP NSELECTED SELECT will give a top 10.
ALLMEMBERS	Determines n from all members in the dimension. For example, if Product has 50 members, then: ORDER PRODUCT ON SALES TOP ALLMEMBERS SELECT will give a top 50.
CONCENTRATE	CONCENTRATE is only valid if it is used in conjunction with, and immediately follows, TOP BOTTOM n on the command line.
[CONCENTRATE <options>]	<p>The <options> parameter can be one or more of the following list: [OtherName COUNT ALLVARIABLES SUPPRESS {NONE BOTH MISSING ZEROS}]*</p> <p>OtherName - a quoted member name to contain the subtotal of all members omitted from TOP BOTTOM n. The default name is 'All Other'.</p> <p>Note: If the specified OtherName already exists in the dimension, a unique name is created by appending '\$' characters to the end of the name.</p> <p>COUNT - generates a count of the number of members included in the OtherName member. The count displays in parentheses () at the end of the long name of the OtherName member, and it is written to a control variable called COUNT8020.</p> <p>ALLVARIABLES - the remainder subtotal is computed for all selected variables, not just those in the ORDER ON clause.</p> <p>SUPPRESS - excludes some combination of missing or zero data from the remainder total and count. By default, missing data is not suppressed.</p>
<member criteria>	Currently supports the keyword EVERYMEMBER, which uses all members of the ORDER dimension (rather than the selected members) to evaluate the TOP BOTTOM <n>.
<members>	Names of one or more members separated by commas. Any members not specified follow in their default order.
SELECT	Automatically selects just the members being ordered.
ACROSSDOWN	Use this keyword if you have attributes selected and included in your across/down list, or if the dimension that you are ordering is an attribute dimension. ACROSSDOWN indicates that the data contained in the model's

ORDER

attributes needs to be realized before the ORDER is performed. Otherwise, the attribute's result member will be used, even if the result is not selected. If your dimensional model (or the current view) does not contain attributes, this keyword is ignored.

Default order

The default order for LIST and DISPLAY is the order in which members occur in the dimension.

Notes:

The statement "ORDER <dimension> ON var TOP <n> SELECT" only operates on the currently selected members of <dimension>. For example: you have a dimension REGION with 16 members. If only seven members of REGION are selected, "ORDER REGION ON SALES TOP 10 SELECT" will only order the seven selected members. However, if the entire 16 members of the dimension are selected, this command will select the 10 members of REGION with the highest SALES value and arrange them in descending order.

When you order a dimension on a variable, your view needs to be narrowed down to a single column of information. You need to make sure that you have only one member of the dimensions that you are not ordering selected. In addition, you should order on a single time period. You can add a period and a periodicity to your ORDER statement. For example:

```
SET PERIOD 99
SET LATEST JULY 99
ORDER Product ON Sales YTD
```

or

```
ORDER Product ON Sales MONTHLY PERIOD JULY 99
```

Example

...This example orders a dimension and selects the ordered items at the same time. When you use ... subsequent time conversions, for example, LIST YTD, selection has an ...important impact on efficiency:

```
ORDER CHANNEL DIRECT, WHOLESALE SELECT
```

...This example orders the top 50 selected Customer members ...based on sales. Application Server only lists the top 50 in the output.

```
ORDER CUSTOMER ON SALES TOP 50
```

...A variable Sales is dimensioned by a dimension Store. To see ...sales displayed from largest to smallest sales figures, enter the following:

```
ORDER Store ON Sales
```

...To see sales displayed alphabetically, enter the following:

```
ORDER Store ALPHABETIC
```

...This example orders and selects the top 10 product members based on sales and lists the top 10 in ... the output. The time values used for each product are the LAST time period in the rows that result. ... The rows that result are:the selected members if only one member was selected ... (of the other dimensions not in the across/down list) the result members if all members were ... selected the first selected member if some, but not all, members ...were selected.

```
ORDER PRODUCT ON SALES TOP 10 SELECT
```

...The dimension in this example contains multiple hierarchies. This example orders the specified ...dimension by all hierarchies, and selects all members of all hierarchies in the dimension:

ORDER MHDIM HIERARCHY ALLHIERARCHIES

SELECT MHDIM ALL

ACROSS TIME,VAR DOWN MHDIM

SEL SALES

LIST

...The output displays all members of all hierarchies, wherever they occur. For example, ONE occurs
... in ODD, LOW, and THREELETTERS, and EIGHT occurs in EVEN, HIGH, and FIVELETTERS:

	Jan 99
	SALES
TOTAL MHDIM	55.00
ODD	25.00
ONE	1.00
THREE	3.00
FIVE	5.00
SEVEN	7.00
NINE	9.00
EVEN	30.00
TWO	2.00
FOUR	4.00
SIX	6.00
EIGHT	8.00
TEN	10.00
TOTAL HIGHLOW	55.00
LOW	15.00
ONE	1.00
TWO	2.00
THREE	3.00
FOUR	4.00
FIVE	5.00
HIGH	40.00
SIX	6.00
SEVEN	7.00
EIGHT	8.00
NINE	9.00
TEN	10.00
TOTAL LETTERS	55.00
THREELETTERS	19.00
ONE	1.00
TWO	2.00
SIX	6.00

ORDER (Report)

TEN	10.00
FOURLETTERS	18.00
FOUR	4.00
FIVE	5.00
NINE	9.00
FIVELETTERS	18.00
THREE	3.00
SEVEN	7.00
EIGHT	8.00

6.122 ORDER (Report)

Use

ORDER is a Report command that defines the order in which columns appear on the page.

Syntax

SET

. ORDER <columns>

ENDSET

Note: The first ORDER statement must appear within the SET-ENDSET block.

Parameter	Description
<columns>	<p>One or more column numbers, ranges of columns, or expressions, separated by commas. The following rules apply to this field:</p> <p>Ranges can be in sequential or reverse sequential order.</p> <p>Application Server numbers the columns in a report sequentially starting from 0. Column 0 contains the name of the row. Column 1 is the first column that appears in the default order, column 2 is the second, and so on. Any column number can be repeated, and any column number or numbers can be omitted.</p> <p>Tip: Use SHOW ACROSS DETAIL to display the order of currently selected columns.</p> <p>Expressions evaluate for each row statement in the report. The left side of the equation is a scalar quantity. The right side of the equation is an expression. The right side of the equation can also include references to columns with the notation c_i, where i refers to the ith column of the report. An equation entirely enclosed in parentheses evaluates, but does not print; use the left side in an expression that you want to print.</p> <p>To use labels as column headings, place the labels, enclosed in double quotation marks (" "), after the name on the left side of the equal sign.</p> <p>Any label may be applied to a column by placing the label after the calculated column name and before the equals sign. For example, ORDER 0-3,PCT '% of Sales' = c1 % c2.</p>

Rules

Application Server normally calculates the column calculations specified with ORDER for every printed row. You can suppress these calculations for certain rows by enclosing the rows in NOCALC-ENDNOCALC statements.

You can include the SHORT keyword in an ORDER statement to specify that short names should appear instead of labels for column 0. You can either substitute 0 with the SHORT keyword, or if

you want to display both the short names and labels, include both SHORT and 0 in the ORDER statement.

Example

...Assume the columns in your report are in the following order:

PRODUCTS TV					
	Jan 00	Feb 00	Mar 00	Apr 00	May 00
Sales	12,200	13,500	14,409	15,355	15,721
Expenses	6,308	6,100	7,147	7,439	7,859

...This block:

```
SET
ORDER 1-3, 0, 'Pct' = c4 % c5, 5-4
ENDSET
```

...produces the following report:

PRODUCT TV						
	Jan 00	Feb 00	Mar 00		Pct	May 00 Apr 00
	12,200	13,500	14,409	Sales	98	15,721 15,355
	6,308	6,100	7,147	Expenses	95	7,859 7,439

6.123 OUTPUT

Use

OUTPUT sends all output that normally appears on your screen (except errors) to a specified destination. When you use OUTPUT with a document or database, the default is for Application Server to overwrite the file.

Syntax

```
OUTPUT {TERMINAL}
      {[LOCAL] PRINTER}
      {GRAPHICS <device>}
      {<document> [;<database> | ;EXTERNAL] [APPEND]}
      {OFF}
```

Parameter	Description
TERMINAL	Displays output on your screen.
PRINTER	Sends output to your default system printer.
LOCAL PRINTER	Sends your output to the client computer printer.
GRAPHICS	Sends output to a graphic device such as a terminal, plotter, printer, or graphics adapter card, supported by Application Server.
<device>	Name of the graphic device on your system.
<document>	Name of an Application Server document to which you sent your output.
<database>	Name of the database where the document is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the document is a text file that is not in an Application Server database. If the document is a DOS file, its name cannot have an extension.
APPEND	Overrides the default and adds the output to the end of the document.
OFF	Turns off the previous OUTPUT command and sends all subsequent output to your terminal.

OUTPUT (Dimension)

Example

...This example prints BUDGETREPORT:

OUTPUT PRINTER

DISPLAY budgetreport

OUTPUT OFF

6.124 OUTPUT (Dimension)

Use

OUTPUT is a Dimension command that defines the intermediate, consolidated members of a dimension. The *output members* represent the consolidated groups of input members or lower level output members.

Syntax

OUTPUT(<number>)[...<hierarchy> HIERARCHY]

. <member> ['<label>']

Parameter	Description
<number>	<p>Number of the hierarchy that the output members belong to. The first OUTPUT(1) statement is the first output level in the first hierarchy. The next OUTPUT(1) statement is a higher output level that is part of the same hierarchy.</p> <p>The first OUTPUT(2) statement is the first output level in the second hierarchy, and so on.</p> <p>Note: If the dimension has only one hierarchy, Application Server identifies all output levels as OUTPUT(1).</p> <p>If the dimension has multiple hierarchies, the first top you define appears in the RESULT statement in the dimension set. Any other tops appear as the final output statements for their hierarchies.</p>
<hierarchy>	<p>Name of the hierarchy as defined in the CONSTRUCT <dimension> HIERARCHY <hierarchy> statement. A hierarchy name appears only when you define multiple hierarchies for the dimension.</p>
<member>	<p>Name of the output member, up to 96 bytes (including periods (.) and underscores (_)). The parameter <member> must begin with a letter. To include a space or a special character in a member name, enclose the name in single quotation marks (' ').</p> <p>The OUTPUT section can contain multiple members.</p>
'<label>'	<p>A text string to assign to the output member, enclosed in single quotation marks. A label is a long description of the output member name. A label can be up to 250 characters long, but is recommended to not exceed 30 characters.</p> <p>Note: The members that you define in an OUTPUT statement are called <i>output members</i>. Member names and labels are sometimes referred to as short names and long names. When displaying data, use SET SHORT LONG to display the member names or labels. The '<label>' option is equivalent to using the LABEL keyword in a CONSTRUCT command.</p>

Rules

You must use unique member names. You do not have to use unique member labels, but they are recommended.

- A dimension need not have output members.

You can enter data at the output level, if applicable.

- An OUTPUT statement appears directly after the INPUT statement and before any HIERARCHY, RESULT, or consolidation statements.
- You specify the OUTPUT statement one or more times, depending on the consolidation groupings. Each OUTPUT statement is followed by a list of member names and, optionally, labels. Except for the last line, <name> '<label>' statements are separated by a comma.

Example

...This example shows cities at the input level that are consolidated into states, the output level.

INPUT LA 'Los_Angeles', San_Francisco, Las_Vegas, Seattle, Yakima

OUTPUT CA 'California', Nevada, Washington

CA = SUM LA, San_Francisco

Nevada = SUM Las_Vegas

Washington = SUM Seattle, Yakima

CA = SUM LA, San_Francisco

Washington = SUM Seattle, Yakima

6.125 PARHEADING-ENDPARHEADING (Report)

Use

PARHEADING-ENDPARHEADING is a Report command that defines a block of headings that appear before each new combination of members.

Syntax

```
PARHEADING
.
.      <heading specifications>
.
ENDPARHEADING
```

Example

...If you enter:

ACROSS Time DOWN Country, Product, Variables

...the default PARHEADING would list the complete country and product name before the table of data.

... For example, Country US, Product TV, and so on.

...The following block displays only the product and country names, for example, US TV:

PARHEADING

TEXT <Country>, <Product>

ENDPARHEADING

6.126 PEEK (Access)

Use

PEEK is an Access command that displays data in an external source before reading it. You can use this command to verify that you have properly defined the data in the external source with ACROSS/DOWN and SELECT commands.

PERCHG()

Syntax

```

PEEK
  [ARRAYSIZE <n>]
  [CREATED]
  [LOGIC <logic>]
  [NONUMBER]
  [NOHEADERS]
  [ONLY <integer>]
  [SKIP <integer>]
  [TABS '<character>']

```

Parameter	Description
ARRAYSIZE <n>	<p>Specifies a number between 1 and 32,768 for the size of the array you want to fetch from the RDBMS during the peek process. If you do not specify an array size, rows are fetched in batches of 100.</p> <p>Note: Some drivers do not support array fetching, such as the Microsoft ODBC driver for MS Access. In those cases, the array size will have a value of 1, and Link will only run single row fetches.</p> <p>Note: If you do not specify the ARRAYSIZE keyword on the command line or you do not add an ARRAYSIZE=n parameter in the <i>[linkid]</i> section of the LSDAL.INI file, the default array size value is 100. If you add the ARRAYSIZE parameter to LSDAL.INI, that value will be used instead of the default value. If you use the ARRAYSIZE keyword on the PEEK command, the PEEK ARRAYSIZE command will override all default values and ARRAYSIZE parameter settings in LSDAL.INI.</p> <p>Tip: The best arraysize may be different on different systems and networks, so you should experiment with arraysize numbers until you find the optimal value.</p>
ONLY	<p>Displays only the specified number of records (rows).</p> <p>Note: ONLY is unnecessary, but its inclusion does not cause an error.</p>
<integer>	Number of records (rows).
SKIP	Skips over the specified number of records (rows) before displaying data.
CREATED	Displays data that Application Server defined using CREATE.
LOGIC	Executes a specified logic set as Application Server examines the external data.
logic	Name of a logic set.
TABS	Displays data using tabs between fields.
character	Indicates the character to be used between fields.
NONUMBER	Does not display line numbers.
NOHEADERS	Does not display column headers.

Example

...This example displays records 101-103 in the external data source:

```
PEEK SKIP 100 ONLY 3
```

6.127 PERCHG()

Use

PERCHG returns the percentage of change between the current period and the same period a specified interval (for example, a year or month ago).

Syntax

```
CALCULATE <result> = PERCHG(<variable> [, period])
```

Parameter	Description
<result>	Name of the result variable.

<variable> Name of the variable to evaluate. Variable names that use special characters should be in single quotation marks (' '). **Note:** Virtual variables calculated with time based functions like LAG, LEAD, MOVING, MOVING2, and PERCHG should be used with caution. These measures can be displayed with Application Server's standard and to-date time periodicities but errors will result when a time set is used in place of time.

period Period in which to evaluate: year (default), quarter, month, or week.

Example: PERCHG

...This example returns the percentage of change in Sales

...between the present period and the same period one year ago:

CALCULATE pctchange = PERCHG(Sales)

6.128 PERIODSAVAILABLE()

Use

PERIODSAVAILABLE returns the number of periods available using the current SET PERIOD start date.

Syntax

CALCULATE <result> = PERIODSAVAILABLE(<variable> ,<when>)

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable to evaluate in double quotes.
<when>	Number from 0 to 4 as follows: when0 – returns the number of periods available to the latest data available for the variable from the SET PERIOD start date when1 – returns the number of periods available to the LATEST period from the SET PERIOD start date when2 – returns the number of periods available to the SET PERIOD start date from the start of the fiscal year when3 – returns the number of periods available to the end of the fiscal year from the SET PERIOD start date when4 – returns the number of periods available to the number of periods per year for this variable

6.129 PREFACE (Report)

Use

PREFACE is a Report command used with applications that defines the number or character to use for input, output, or result dimension members. Application Server uses it for hierarchical drill-down purposes.

Note: The PREFACE command returns values even if the member names or labels are not displayed. Two sets of preface characters, one on the short names and one on the labels, are returned if both are displayed.

You can use the RANK keyword to display the sorted position within the level, in either the SORT column or any other non-calculated column.

Syntax

PREFACE | '123' | RANK [NLEVEL <n>] [<col>]

PRINT

Parameter	Description
1	Indicates an input member.
2	Indicates a result member.
3	Indicates an output member.

Notes:

- You can substitute characters for 123, for example, abc. However, if you use numbers in your application, you can base format colors on them.
- The PREFACE flag appears as the first character of a row name. You can use TABS to tab-delimit the flags and row names.

RANK [NLEVEL <n>] [<col>]

Specifies the nested level to use. The optional parameter <col> specifies the column to rank on. If you do not specify a value for <col>, the SORT column is used by default.

Note: The PREFACE RANK flag is followed by a single space.

Example: PREFACE RANK

...This Syntax:

SORT 2

SORT NLEVEL 2 1

PREFACE RANK 1

PREFACE NLEVEL 2 1

...produces the following output:

	Sales	Sales	Market %
Market Total	18,410,027.10	19,232,828.99	100.00
Direct	15,706,322.95	16,402,453.14	85.31
1 Grapevine Goodness 12oz Conc	2,024,328.17	2,157,967.50	11.00
6 Market Basket - Chelmsford,	38,859.84	45,041.00	0.21
12 Winn-Dixie - Miami, FL	36,604.80	44,027.00	0.20
4 Walgreens - Loveland, CO	39,579.84	43,924.00	0.21
19 RITE-AID - Worcester, MA	35,452.80	43,047.00	0.19
1 Jewel - Wheaton, IL	41,891.52	42,798.00	0.23
76 7 Eleven - Nashua, NH	8,094.24	9,577.75	0.04
Total Printed (6)	200,483.04	228,414.75	1.09
Total Other (77)	1,823,845.13	1,929,552.75	9.91

6.130 PRINT

Use

PRINT sends a listing of one or more sets to a printer.

Syntax

PRINT [<settype>] <setnames> [LOCAL]

Parameter	Description
<settype>	Type of set: dimension, document, logic, procedure, report, synonym, or time. If you do not specify a set type, Application Server uses the first set found with the specified name.

<setnames> One or more set names that you want to send to the printer, separated by commas.

LOCAL Sends the printout to the client machine's printer.

Example

...This example sends the dimension **Country** to the printer:

PRINT dimension Country

...This example sends the dimensions **Country, Type, and Product** to the printer:

PRINT dimension Country, Type, Product

6.131 PROCEDURE

Use

PROCEDURE starts the Procedure editor, where you can create or edit procedures.

Syntax

PROCEDURE <procedure> [;<database> | ;EXTERNAL|LOCAL]

Parameter	Description
<procedure>	Name of a new or existing procedure. If you do not specify a name, Application Server uses the default procedure if you defined it, or the last procedure you edited.
<database>	Name of the database where the procedure is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the procedure is a text file that is not in an Application Server database. If the procedure is a DOS file, its name cannot have an extension.
LOCAL	The text file is a client file.

Notes:

You cannot use command separators in the Procedure editor.

If you create a procedure in an attached database while running in CHECKPOINT FREEZE, you cannot obtain a listing of that procedure by entering EXHIBIT PROCEDURE FROM <database> or DIR DATA <database> PROCEDURE until you perform a CHECKPOINT UPDATE.

6.132 PURGE

Use

PURGE deletes all backup copies of sets in your Use database or in a specified database.

Application Server creates backup copies of all sets that have been edited or removed since the last PURGE or EXIT CLEAR. You can recover a set as it was before you last edited it when you have backup copies.

If a database becomes nearly full, some commands that require temporary space in the database might not have enough space to execute. Purging the database frees space.

Note: If you enter PURGE, you cannot recover versions of sets edited before PURGE.

Syntax

PURGE [<database>]

Parameter	Description
<database>	Name of the database to purge.

QUIT**Example: PURGE**

...This example shows that PURGE has removed the backup (recoverable) version of the report ... WEEKLY. The Size field indicates the number of blocks the report occupies.

DIRECTORY reports FULL

...The output of this command would be:

	RecoverSize	Last Updated
REPORTS		
WEEKLY R	2	Thu Sep 5 15:20:16 2000
MONTHLY	2	Wed Sep 4 11:00:50 2000
QUARTERLY	2	Wed Aug 28 16:07:48 2000

PURGE finance

DIRECTORY reports FULL

...The output of this command would be:

	RecoverSize	Last Updated
REPORTS		
WEEKLY	2	Thu Sep 5 15:20:16 2000
MONTHLY	2	Wed Sep 4 11:00:50 2000
QUARTERLY	2	Wed Aug 28 16:07:48 2000

6.133 QUIT

Use

QUIT exits from Application Server without saving any new data. It discards the values of all temporary variables and restores the database to the state it was in when Application Server last updated it. Usually, this state is the point after the last command Application Server completed without an error.

Syntax

QUIT

Example: QUIT

...This example shows what happens when you try to enter EXIT CLEAR

...without specifying whether you want to save the data in your temporary variables:

EXIT CLEAR

...Application Server issues this message:

Either Must SAVE DATA and then EXIT CLEAR or, EXIT, or QUIT.

...If you now enter:

QUIT

...You return to the host operating system; temporary variables are not saved.

6.134 RANDOM()

Use

RANDOM returns a random number greater than or equal to 0 and less than or equal to 1. You can use RANDOM to generate test data for a variable.

Syntax

CALCULATE <result> = RANDOM([<number>]) <expression>

Parameter	Description
<result>	Name of the result variable.
<number>	Optional constant number between 0 and 1. Application Server generates the same random numbers for the same values of <number>.
<expression>	Any mathematical expression. For example, RANDOM(0.5) * 100.

Example

...This example creates random data for a test variable, testcogs. The input levels of dimensions are ... selected, the variable is selected, and then the random data is generated with the FULL keyword so ...that data is not consolidated.

CREATE testcogs LIKE cogs

SET PERIOD 2000

SELECT Product input

SELECT Region input

SELECT Type input

SELECT testcogs

ACROSS time DOWN variables

CALCULATE testcogs=random(.6)*1000 FULL

LIST monthly PERIOD jan 00 - may 00

...The output of this command would be:

	Jan 00	Feb 00	Mar 00	Apr 00	May 00
TESTCOGS	831	969	687	395	945

6.135 RANK()

Use

RANK returns the order of dimension members established by ORDER. For dimension members, there is always a default order (1, 2, 3, ...) that provides a default value for rank, even if you have not entered ORDER.

Syntax

CALCULATE <result> = RANK(<dimension>)

Parameter	Description
<result>	Name of the result variable.
<dimension>	Name of the dimension that Application Server has ordered with ORDER.

READ (Access)

Example

...This example orders the members of the dimension Merchandise based on the decreasing values of ... Sales. The ordering is then preserved in the variable Sales_Rank.

ORDER Merchandise ON Sales

CALCULATE Sales_Rank = RANK(Merchandise)

...You can now use Sales_Rank in a comparison with last year's sales rankings.

6.136 READ (Access)

Use

READ is an Access command that reads from an external source. You can read data into either an input level or an output level of a constructed dimension.

Notes about optimizing READs

ACCESS READ works best if external data is sorted by dimensions alphabetically.

You will achieve the best performance when TIME is ACROSS. With TIME across, Application Server doesn't have to do any pivoting; it can write the series out as it appears in the external source with minimal effort.

If TIME is DOWN, you will achieve the best performance when TIME is the fastest varying field in the external source and you also specify the FASTMODE keyword. Then Application Server can accumulate series from consecutive records without too much effort. FASTMODE gives you a big in-memory buffer for the uncompressed series. The entire series is built up in memory in the order they appear and when the hash table is full, they get flushed out as whole series.

If TIME is DOWN, and TIME is the slowest changing field, this is the worst case you can have. It doesn't matter whether you use the FASTMODE keyword in this case. This format forces Application Server to perform excessive reading, rereading, uncompressing, recompressing, rewriting, buffer scanning, and buffer reorganization and the series start to get scattered around blocks.

Syntax

```
READ
  [ADD | SUBTRACT]
  [ARRAYSIZE <n>]
  [ERRORS <file>]
  [FASTMODE [n] [FLUSH]]
  [LOGIC <logic>]
  [NOFASTMODE]
  [NOFLUSH]
  [ONLY <integer>]
  [SAVE <integer>]
  [SKIP <integer>]
  [SORTED]
  [STOP <integer>]
  [UPDATE]
  [ZEROISNULL]
```

Parameter	Description
ADD	Adds the external values to the existing values. The default replaces existing values with those read in.
SUBTRACT	Subtracts the external values from the existing values. The default replaces existing values with those read in.

ARRAYSIZE <n>	<p>Specifies a number between 1 and 32,768 for the size of the array you want to fetch from the RDBMS during the read process. If you do not specify an array size, rows are read in batches of 100.</p> <p>Note: Some drivers do not support array fetching, such as the Microsoft ODBC driver for MS Access. In those cases, the array size will have a value of 1, and Link will only run single row fetches.</p> <p>Note: If you do not specify the ARRAYSIZE keyword on the command line or you do not add an ARRAYSIZE=n parameter in the <i>[linkid]</i> section of the LSDAL.INI file, the default array size value is 100. If you add the ARRAYSIZE parameter to LSDAL.INI, that value will be used instead of the default value. If you use the ARRAYSIZE keyword on the READ command, the READ ARRAYSIZE command will override all default values and ARRAYSIZE parameter settings in LSDAL.INI.</p> <p>Tip: The best arraysize may be different on different systems and networks, so you should experiment with arraysize numbers until you find the optimal value.</p>
ERRORS	Copies records that are not read to a specified file. Use ERRORS only when the ACCESS subsystem is EXTERNAL.
<file>	Name of the file.
FASTMODE [<n>]	<p>Allocates a large in-memory buffer, holding up to <n> time series in memory, to optimize performance of I/O during the READ. By default, if you do not specify FASTMODE, then FASTMODE 100000 is used.</p> <p><n> is determined by n is determined by:</p> $\text{minimum}((\text{product of number of selected members from each dimension in the ACROSS and DOWN}) + 10, 100000)$ <p>Notes:</p> <p>Make sure you have a SET MEMORY with a suitable value. The FASTMODE buffer allocates 8 bytes per time period. So for example, if you have monthly data and SET PERIOD for 12 months, and <n> is 100000, it will allocate at least 19200000 bytes just for the FASTMODE buffer, plus any other memory that is used for other reasons. If you have a small SET MEMORY setting, the command will fail. You must either set a larger SET MEMORY value or set a smaller FASTMODE or use NOFASTMODE to turn this feature off.</p> <p>For small databases, the FASTMODE keyword does not make any difference.</p> <p>If you know that the external source contains many fewer series than <n>, you may want to set a smaller value of <n> to use less memory or use NOFASTMODE to turn this feature off.</p> <p>If you know that the external source has a very large number of records, you may gain performance improvements by specifying a larger value of <n>, say 1 million. However, note that the operating system is unlikely to allocate Application Server very large amounts of memory and in practice a 32 bit process won't be allowed to allocate more than say 1GB of memory.</p> <p>Use NOFASTMODE and NOFLUSH keywords to turn this functionality off.</p>
FLUSH	Flushes Application Server's buffer cache (the cache of database pages set by SET BUFFERS) when it fills.
LOGIC	<p>Executes the specified logic set as Application Server reads each record. In this way, you can modify values read before updating variables. Logic statements operate on one record of the table at a time. Data outside the record (for example, in variables) is not available for use in these calculations.</p> <p>Note: A logic which calculates the missing values to zero does not execute when used with the READ command. Use the CALCULATE command in these cases.</p>
<logic>	Name of the logic set.
NOFASTMODE	Turns off the FASTMODE functionality.
NOFLUSH	Turns off the FLUSH functionality.
ONLY	<p>Limits the data read to the specified number of records.</p> <p>Note: READ ONLY is the default. ONLY is unnecessary, but its inclusion does not cause an error.</p>

READ (Access)

<integer>	Number of records.
SAVE	Temporarily saves the data read in your Use database, every specified number of records. The default saves the data every 2000 records.
SKIP	Skips the specified number of records before reading data.
SORTED	Indicates data has been previously sorted by a dimension. If this dimension is the last down dimension, Application Server reads the data faster.
STOP	Stops reading data if Application Server skips more than the specified number of records due to errors.
UPDATE	Updates the database each time Application Server saves records. The default updates the database only after Application Server reads all the records.
ZEROISNULL	Filters zero values from the external source and treats them as missing in the Application Server database.

Example

...Read all data in the external data source:

READ

...Read the data and adds the values read into existing variables, stopping if there are any errors:

READ ADD STOP 1

...Copy records that are not read to the file ACCTERR:

READ ERRORS accterr

...Construct dimensions, compile them, create variables, and then reads in data from DRINKS.DBF.

... The .DBF extension is assumed. The data is used to create the dimensions

...Product, Channel, and Region, and the variables Actsales and Budsales.

ACCESS dbase

USE drinks

CONSTRUCT product level product

COMPILE dim product

CONSTRUCT channel level channel

COMPILE dim channel

END

CREATE var actsales by product, channel

CREATE var budsales by product, channel

SELECT var

SET PERIOD Jan 94 - Jan 96

ACROSS variables DOWN product, channel, time

ACCESS dbase

USE drinks

READ

END

Reading data into the attribute variables

Note: Use this procedure as a syntax example - it will not work with the demonstration databases in your installation.

...Select the attribute variables and then read in the data

SELECT var pkg_type, size

```

SELECT drinks input
ACROSS var DOWN Time, Drinks
ACCESS LSLINK
CONNECT dbase
SELECT * FROM Drinks
PEEK ONLY 5
LSS CREATE size = size
SELECT * FROM Drinks
...Read data into the variable
READ
END
...This shows the data that was loaded into the attribute variables:
      PKG TYPE  HIER ATT
COLA  BOTTLE32_OZ.
SPRITZ CAN    8_OZ.

```

6.137 RECONSTRUCT (Access)

Use

RECONSTRUCT is an Access command that reorders the output members in a dimension. You only use this command if you receive a message that you cannot use a dimension in the Rollup subsystem.

Syntax

RECONSTRUCT <dimension>

Note: You must compile a dimension after you have reconstructed it.

Parameter	Description
<dimension>	Name of a compiled dimension.

Example

...To reorder the output members of the PRODUCT dimension, enter the following statements:

```

ACCESS EXTERNAL
RECONSTRUCT Product
END
COMPILE dimension Product

```

6.138 RECOVER

Syntax

RECOVER [<settype>] <name>

Use

RECOVER restores the previous version of a set from the current session.

RECOVER returns the set to the condition it was in when you entered Application Server after an EXIT CLEAR or QUIT, or to the initial version if you created the set in the current session.

REMOVE

Note: If you enter PURGE, you cannot recover versions of sets edited before PURGE.

Parameter	Description
<settype>	Type of set: dimension, document, logic, procedure, report, synonym, or time. If you do not specify a set type, Application Server uses the first set it finds with the specified name.
<name>	Name of the set to recover.

Note: RECOVER does not recover the compiled form of a dimension; a database reorganization is not forced. You must enter COMPILE to force the database reorganization.

Example: RECOVER

...This example recovers the dimension BUDGET:

RECOVER dimension budget

6.139 REMOVE

Use

REMOVE deletes sets, variables, database versions, formats, or data.

Syntax

REMOVE

```
{<settype> <setname> [..., <setname>] [SURE] }
{ ADIMENSION <attribname> }
{ ATTRIBUTE <attribname> }
{ CONSOLIDATED <variable> [..., <variable> ] [ UPDATE <n> ] [ BEFORE <date> ] [ AFTER <date> [
NOTRUNCATE ] ] [ SURE ] }
{ CUSTOM [<database>.] [<owner>.]<dimension>.<setname>
[,<database>.] [<owner>.]<dimension>.<setname>... ] }
{ DATA <variable> [..., <variable> ] [ UPDATE <n> ] [ BEFORE <date> ] [ AFTER <date> [
NOTRUNCATE ] ] [ SURE ] }
{ FORMAT <variable> [..., <variable>] }
{ ROLLUPS }
{ SELECTED <variable> [..., <variable> ] [ UPDATE <n> ] [ BEFORE <date> ] [ AFTER <date> [
NOTRUNCATE ] ] [ SURE ] }
{ VARIABLE <variable> [..., <variable> ] [FROM remotedb] [TEMPORARY] }
{ VERSION <version> [..., <version>] }
```

Parameter	Description
<settype>	Type of set: dimension, document, logic, procedure, report, synonym, or time.
<setname>	Removes one or more set names of the specified set type, separated by commas. You can use an asterisk (*) in the name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name.
SURE	Applies to dimension sets only. Forces removal of the compiled dimension set.
ADIMENSION <attribname>	Removes the compiled attribute that results from compiling the attribute set.
ATTRIBUTE <attribname>	Removes an attribute set that describes the attribute.
CONSOLIDATED <variable>	Removes all output level data for the specified variable, including input data loaded at an output level. Input level data is untouched. You can use an asterisk (*) in the name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name.
UPDATE <n>	Forces Application Server to update the database after removing data for <n> time series. This helps minimize the space required for processing when removing data.
BEFORE <date>	Removes data for the variable(s) before the specified date.

AFTER <date>	Removes data for the variable(s) after the specified date. Note: If BEFORE and AFTER are used together, a slice of data in the middle of the timeseries is removed. For example, if you use the command REMOVE DATA BEFORE 8/2006 AFTER 6/2006, you remove data for 7/2006. If you use the command REMOVE DATA BEFORE 6/2006 AFTER 8/2006, you remove all data except for 7/2006.
NOTTRUNCATE	Updates the index to the data without actually removing the data. Similar to the REMOVE DATA AFTER date command, data specified after the AFTER date will no longer appear or be available. While the REMOVE DATA AFTER command actually removes the data, REMOVE DATA AFTER NOTTRUNCATE does not remove the data, making the data removal process much quicker. The index is modified to amend the time span corresponding to the series. When you have a lot of data removal, this keyword improves performance without any visual differences from REMOVE DATA AFTER. For example, if you use REMOVE DATA AFTER 6/2006 NOTTRUNCATE, data for 7/2006 and onward will no longer appear or be available. Notes: If BEFORE and AFTER are used together, you cannot use the NOTTRUNCATE keyword with AFTER.
SURE	Removes data for the listed variable. SURE is required when using an asterisk (*) in a name. If you do not use SURE, the system produces an error unless you enter CHECKPOINT FREEZE.
CUSTOM	Removes a saved User-Defined Hierarchy and the compiled dimension set that was saved during a SAVE CUSTOM command. At a minimum, specify the dimension containing the User-Defined Hierarchy, and the procedure set name of the User-Defined Hierarchy that was created during a SAVE CUSTOM command. Notes: You can remove more than one User-Defined Hierarchy at a time by specifying each one separated by a comma. To remove a User-Defined Hierarchy, you must be either the administrator, or the owner of the User-Defined Hierarchy.
<database>	Name of the database containing the User-Defined Hierarchy to be deleted. If you omit the database name, the current USE database is used.
<owner>	Name of the Application Server user who is responsible for the User-Defined Hierarchy. This name was specified during the SAVE CUSTOM command when the User-Defined Hierarchy was created. If you omit the owner name, the current Application Server user name is used.
<dimension>	Name of the dimension whose User-Defined Hierarchy you want to remove.
<setname>	Name of the procedure set where the User-Defined Hierarchy was saved during the SAVE CUSTOM command.
DATA <variable>	Removes data for the specified variable(s). Application Server updates the database after it has modified all the relevant time series. You can use an asterisk (*) in the name to indicate that any character can occupy that position or any of the remaining positions in the name. You must use SURE when specifying an asterisk (*).
UPDATE <n>	Forces Application Server to update the database after removing data for <n> time series. This helps minimize the space required for processing when removing data.
BEFORE <date>	Removes data for the variable(s) before the specified date.
AFTER <date>	Removes data for the variable(s) after the specified date. Note: If BEFORE and AFTER are used together, a slice of data in the middle of the timeseries is removed. For example, if you use the command REMOVE DATA BEFORE 8/2006 AFTER 6/2006, you remove data for 7/2006. If you use the command REMOVE DATA BEFORE 6/2006 AFTER 8/2006, you remove all data except for 7/2006.
NOTTRUNCATE	Updates the index to the data without actually removing the data. Similar to the REMOVE DATA AFTER date command, data specified after the AFTER date will

REMOVE

no longer appear or be available. While the REMOVE DATA AFTER command actually removes the data, REMOVE DATA AFTER NOTRUNCATE does not remove the data, making the data removal process much quicker. The index is modified to amend the time span corresponding to the series. When you have a lot of data removal, this keyword improves performance without any visual differences from REMOVE DATA AFTER.

For example, if you use REMOVE DATA AFTER 6/2006 NOTRUNCATE, data for 7/2006 and onward will no longer appear or be available.

Note: If BEFORE and AFTER are used together, you cannot use the NOTRUNCATE keyword with AFTER.

SURE	Removes data for the listed variable. SURE is required when using an asterisk (*) in a name. If you do not use SURE, the system produces an error unless you enter CHECKPOINT FREEZE.
FORMAT <variable>	Removes all formatting that Application Server assigned to variable(s) with SET VARIABLES. You can use an asterisk (*) in the name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name.
ROLLUPS	Removes all rollup definitions.
SELECTED <variable>	Removes data for the specified variable(s), for all currently selected dimension member combinations.
UPDATE <n>	Forces Application Server to update the database after removing data for <n> time series. This helps minimize the space required for processing when removing data.
BEFORE <date>	Removes data for the variable(s) before the specified date.
AFTER <date>	Removes data for the variable(s) after the specified date. Note: If BEFORE and AFTER are used together, a slice of data in the middle of the timeseries is removed. For example, if you use the command REMOVE DATA BEFORE 8/2006 AFTER 6/2006, you remove data for 7/2006. If you use the command REMOVE DATA BEFORE 6/2006 AFTER 8/2006, you remove all data except for 7/2006.
NOTRUNCATE	Updates the index to the data without actually removing the data. Similar to the REMOVE DATA AFTER date command, data specified after the AFTER date will no longer appear or be available. While the REMOVE DATA AFTER command actually removes the data, REMOVE DATA AFTER NOTRUNCATE does not remove the data, making the data removal process much quicker. The index is modified to amend the time span corresponding to the series. When you have a lot of data removal, this keyword improves performance without any visual differences from REMOVE DATA AFTER. For example, if you use REMOVE DATA AFTER 6/2006 NOTRUNCATE, data for 7/2006 and onward will no longer appear or be available. Note: If BEFORE and AFTER are used together, you cannot use the NOTRUNCATE keyword with AFTER.
SURE	Removes data for the listed variable. SURE is required when using an asterisk (*) in a name. If you do not use SURE, the system produces an error unless you enter CHECKPOINT FREEZE.
VARIABLE <variable>	Removes one or more variable names, separated by commas. You can use an asterisk (*) in the name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name. Variable names that use special characters should be in single quotation marks (' '). Options include: FROM <i>remotedb</i> - Removes a distributed variable from a specified database. TEMPORARY - Removes a temporary variable from the Work database, which was created by the CREATE command.
VERSION <version>	Removes one or more versions of variables created by the CREATE VERSION command. Separate each version with a comma.

Example: REMOVE

...This example removes the logic sets FY00MODEL and FY00BUDGET:

REMOVE logic fy00model, fy00budget

...This example removes data earlier than 2000 for the variable Expenses:

REMOVE DATA Expenses BEFORE 2000

...This example removes all dimension set data. The SURE keyword

...indicates you are sure you want to do this.

REMOVE DATA * SURE

...This example removes an attribute using the following three commands:

REMOVE ATTRIBUTE data

REMOVE ADIMENSION data

REMOVE VARIABLE data

...This example removes all sales data for channel c2 before February 2000:

SELECT CHANNEL c2

SELECT VARIABLE sales

REMOVE SELECTED sales BEFORE feb00

6.140 REMOVE ACCESS (Supervisor)

Use

A Supervisor command that removes an access key from a user record or a database record held in MASTERDB.

Syntax

REMOVE ACCESS <key> {USER <user> | DATABASE <database>}

Parameter	Description
ACCESS <key>	The access key to remove.
USER <user>	The name of the user account from which the access key should be removed. The user will no longer be able to update any databases that have <key> set as an update access key, and the user will no longer be able to read from any databases that have <key> set as a read access key.
DATABASE <database>	The name of the database from which the access key should be removed. If <key> is an update access key, all users will be able to update the database; if <key> is a read access key, all users will be able to read from the database. The database must be marked as available in MASTERDB, and all users must be detached.

Remarks

Before removing an access key, you must ensure that all users are detached from the database.

Example

...This example adds a read access key to the Juice database and to the user named Test1. It then

...removes the access key from the Test1 user, so the Test1 user cannot read from the Juice database.

...Create the Test1 user.

SUPERVISOR CREATE USER Test1 USEDDB Juice

...Add the access key to the Juice database.

REMOVE DATABASE (Supervisor)

SUPERVISOR CHANGE DATABASE Juice READ TestKey

...Add the access key to Test1 so the Test1 user can read from the Juice database.

SUPERVISOR ADD ACCESS TestKey USER Test1

...Remove the access key from Test1 so the Test1 user can no longer read from the Juice database.

SUPERVISOR REMOVE ACCESS TestKey DATABASE Juice

6.141 REMOVE DATABASE (Supervisor)

Use

A Supervisor command that removes a database from MASTERDB, and attempts to delete the database on disk.

Syntax

REMOVE DATABASE <database>

Parameter	Description
DATABASE <database>	The name of the database to delete.

Example: REMOVE DATABASE

...This example removes the database named Demo1. The database is removed from MASTERDB and

...deleted from the disk, so you may want to ensure a backup is available.

SUPERVISOR REMOVE DATABASE Demo1

6.142 REMOVE GROUP (Supervisor)

Use

A Supervisor command that removes security groups.

Syntax

REMOVE GROUP <group>

Parameter	Description
GROUP <group>	The name of the group to remove. Use an asterisk (*) to remove all groups.

Example: REMOVE GROUP

...This example creates a security group named TestGroup, and then removes it.

SUPERVISOR CREATE GROUP TestGroup**SUPERVISOR REMOVE GROUP TestGroup**

6.143 REMOVE PROTECTION (Supervisor)

Use

A Supervisor command that removes a protection key from a database record held in MASTERDB.

Syntax

REMOVE PROTECTION DATABASE <database> <key>

Parameter	Description
<database>	The name of the database from which the protection key should be removed.
<key>	The name of the protection key to remove.

Remarks

By removing a protection key, you enable users to copy or move the database.

Before removing a protection key, you must ensure that all users are detached from the database.

Example: REMOVE PROTECTION

...This example removes a protection key named ProtKey from the Juice database.

SUPERVISOR REMOVE PROTECTION DATABASE Juice ProtKey

6.144 REMOVE USER (Supervisor)

Use

A Supervisor command that deletes a user record from MASTERDB, or removes a user from a security group.

Syntax

REMOVE USER <user> [FROM <group>]

Parameter	Description
USER <user>	One or more user names separated by a comma. If you specify FROM <group>, the user is removed from the security group specified. If you omit FROM <group>, the user is logged out, and the user record is completely removed from MASTERDB.
FROM <group>	The name of the group from which users are removed. An asterisk (*) removes the users from all groups.

Example

...The first part of this example creates two users named Test1 and Test2, and adds them to a security group named TestGroup. The second part of the example removes the user Test1 from TestGroup and removes the user Test2 from MASTERDB.

...Create the users and add them to the security group.

SUPERVISOR CREATE USER Test1

SUPERVISOR CREATE USER Test2

SUPERVISOR CREATE GROUP TestGroup

SUPERVISOR ADD USER Test1, Test2 TO TestGroup

...Remove the user Test1 from TestGroup

SUPERVISOR REMOVE USER Test1 FROM TestGroup

...Remove the user Test2 from MASTERDB

SUPERVISOR REMOVE USER Test2

6.145 RENAME

Use

RENAME changes the name of a specified set or variable.

REPEAT-UNTIL

Note: If you rename variables in a Hybrid OLAP model, you must make the corresponding name changes in the underlying relational database tables.

Syntax

```
RENAME {<settype> | VARIABLE} <oldname> <newname>
```

Parameter	Description
<settype>	Type of set: document, logic, procedure, report, synonym, or time. Note: You cannot rename a dimension set.
VARIABLE	Changes the name of a stored variable. Note: Virtual variables cannot be renamed. Instead, delete the virtual variable and recreate it with the new name.
<oldname>	Current name of the set or variable. Variable names that use special characters should be in single quotation marks (' ').
<newname>	New name of the set or variable. Variable names that use special characters should be in single quotation marks (' ').

Example

...This example changes the name of the logic set REVENUE to BUDREVENUE:

```
RENAME logic revenue budrevenue
```

6.146 REPEAT-UNTIL

Use

REPEAT-UNTIL repeatedly executes a block of statements until a specified condition becomes true.

Syntax

```
REPEAT
    .
    .      <statements>
    .
UNTIL <condition>
```

Parameter	Description
REPEAT	Indicates the start of a loop.
<statements>	Any sequence of statements or commands.
UNTIL <condition>	Indicates the end of a loop, where <condition> is a valid logical condition. If the condition is true, exits from the loop. If the condition is false, executes the statements in the loop.

6.147 REPORT

Use

REPORT starts the Report editor, where you can create or edit reports.

Syntax

```
REPORT [<report>] [<database> | ;EXTERNAL|LOCAL]
```

Parameter	Description
<report>	Name of a new or existing report up to 96 bytes. If you do not specify a name, Application Server uses the default report if you have defined it, or the last report you edited.

<database>	Name of the database where the report is located. If you do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the report is a text file that is not in an Application Server database. If the report is a DOS file, its name cannot have an extension.
LOCAL	Indicates that the report is a client text file.

Notes:

You can create or edit a report with the EXTERNAL or LOCAL extension, but you must copy it into a model and compile it before it can be used in the model.

Use the CTRL-ENTER key combination to force lines of more than 256 characters to be recognized.

Remarks

You can use the following commands within the Report editor:

COLHEADING- ENDCOLHEADING	PARHEADING- ENDPARHEADING	TABS
FOOTNOTE-ENDFOOTNOTE	PREFACE	TEXT
HEADING-ENDHEADING	ROWS	TITLE
MASK-ENDMASK	SET-ENDSET	UDASH
NEWPAGE	SKIP	UDATA
NOCALC-ENDNOCALC	SUBTOTAL-ENDSUBTOTAL	UNAME
NROWS	SUBTOTAL INCLUDE	UTEXT
ORDER	SUMMARY-ENDSUMMARY	

You can use nested levels within reports.

Using nested levels in reports

Using STYLE LEVELTOTAL in SET-ENDSET switches on the feature that allows you to utilize nested levels in a report set. There are $n-1$ (n minus 1) nested levels in a report, where n is the number of Down dimensions. The first Down dimension in a nested level is number 1. There is an additional level, number 0, which contains a total for the entire report. Totals are summed from the selected members and may be displayed or used in calculations in the PARHEADING and/or SUMMARY blocks.

The PARHEADING and SUMMARY specification blocks are executed one or more times depending on the level of the change. For example, when there are four Down dimensions, at the start of the report the PARHEADING block is executed 4 times, with the nested level number set to 0,1,2, and 3. When a change of member occurs in the second Down dimension, the SUMMARY block is executed with the nested level set to 3, then again with the nested level set to 2; the PARHEADING block is then executed with the nested level set to 2, then again with the nested level set to 3.

The LEVELTOTAL style of report is designed to be used when all the selected members in the last Down dimension are the same level (for example, all input members), and the ROWS ALL specification is used to print the report rows. Additional specifications are available to select the actual information which will be displayed within the report.

You can use the following specifications in nested level reports:

Specification	Description
NLevels()	Returns the total number of nested levels.

REPORT

NLevel()	<p>Returns the current nested level number. This is useful when you require conditional processing for each nested level.</p> <p>You can use NLevel() with the ORDER command, for example:</p> <p>ORDER 0,1,'% if NLevel() EQ 1 =c1/Value(NLevelTotal(0),1)*100 otherwise = c1/Value(NLevelTotal(1),1)*100</p>
NLevelName	<p>Gives the current member name within the current nested level.</p> <p>For example:</p> <p>Row NLevelName(nlevel) <NLevelName></p>
NLevelTotal()	<p>Gives the totals for the specified nested level.</p> <p>For example:</p> <p>Row NLevelTotal(nlevel)</p> <p>If you do not specify a parameter, the total for the current nested level is returned.</p> <p>For example:</p> <p>Row NLevelTotal() <NLevelName></p> <p>You can use NLevelTotal with the ORDER command, for example:</p> <p>ORDER 0,1,'% = c1/Value(NLevelTotal(2),1)*100</p> <p>Note: If you use NLevelTotal() in both an ORDER statement and a ROW statement, you should assign the NLevelTotal row to a temporary variable and use that temporary variable in your ROW statement. For example:</p> <p>TEMP NLTOT Summary NLTOT=NLevelTotal() ROW NLTOT <NLevelName> EndSummary</p>
NLevelPrinted()	<p>Gives the totals actually printed for the current nested level.</p> <p>For example:</p> <p>Row NLevelPrinted() "Total Printed"</p>
NLevelOther()	<p>Gives the totals suppressed for the current nested level.</p> <p>For example:</p> <p>Row NLevelOther() "Total Other"</p>

Example 1

...This is an example of a report using NLEVELTOTAL:

```

SET
  across time, var
  down customer, product, channel
  columns once
  tabs
  Style leveltotal
  indent 0, 0, 5, 5
  decimal 0
  ORDER 0,1
  , "%one" if NLevel() EQ 0 = c1/value(NLevelTotal(0),1)*100
  if NLevel() EQ 1 = c1/value(NLevelTotal(1),1)*100
  if NLevel() EQ 2 =c1/value(NLevelTotal(2),1)*100
  otherwise = c1/Value(NLevelTotal(3),1)*100
  SUPPRESS formfeed, heading, missing
ENDSET

parheading
  when nlevel() eq 1
    text <nlevelname
  elsewhen nlevel() eq 2

```

```

        text ' ', <nlevelname>
      endwhen
endparheading
rows all
summary
temp nltot = nleveltotal()
row nltot <nlevelname>
endsummary

```

Example 2

...This is an example of a report using NLEVELPRINTED and NLEVELOTHER:

SET

```

  Style leveltotal
  MISSING ''
...  TABS
...  WIDTH 5,15,8,4
    DECIMAL 2
    COLUMNS ONCE
    ORDER 0,1,
      "% " if NLevel()EQ 0 =c1/value(NLevelTotal(0),1)*100 if NLevel()EQ 1 =c1/value(NLevelTotal(1),1)*100 if
        NLevel()EQ 2 =c1/value(NLevelTotal(2),1)*100 otherwise = c1/Value(NLevelTotal(3),1)*100,
      2,
      "%2" if NLevel()EQ 0=c2/value(NLevelTotal(0),2)*100 if NLevel()EQ 1 =c2/value(NLevelTotal(1),2)*100 if
        NLevel()EQ 2 =c2/value(NLevelTotal(2),2)*100 otherwise = c2/Value(NLevelTotal(3),2)*100
...  SUPPRESS formfeed,heading,missing,parheading
    SUPPRESS formfeed,heading,missing
    INDENT 2,4,6,3 '#'
ENDSET
Heading
  skip 1
...  Text c "<HEADING>"
    Text "&LatestDate - &DateString"
.... Text "</HEADING>"
EndHeading
COLHEADING
...  Text "<COLHEADING>"
    Text "          ", "&Member1", "% Total", "&Member2", "% Total"
...  Text "</COLHEADING>"
ENDCOLHEADING
PARHEADING
...  Text "<PARHEADING ALIGN=LEFT>"
...    when nlevel() eq 1

```

RESTORE CUSTOM

```

                text <nlevelname>
...      elseif nlevel() eq 2
...      text '##', <nlevelname> ...For indenting the parheading
...      endwhen
...  Text "</PARHEADING>"
ENDPARHEADING
rows all
summary
temp nltot = nleveltotal()
row nltot <nlevelname>
row nlevelprinted() <printedcount>
row nlevelother() <othercount>
endsummary

```

6.148 RESTORE CUSTOM

Use

RESTORE restores a previously saved User-Defined Hierarchy from CGLIB.

Syntax

RESTORE CUSTOM <dimension> <owner>.<setname> [APPEND]

Parameter	Description
<dimension>	Name of the dimension or attribute whose saved User-Defined Hierarchies you want to restore for this session. By default, the current USE database is used. You cannot restore a User-Defined Hierarchy that was created in another database.
<owner>	Application Server user responsible for the User-Defined Hierarchy you want to restore. If you omit the owner, the current Application Server user name is used.
<setname>	Name of the procedure set the User-Defined Hierarchy was saved to when it was created and saved with the SAVE CUSTOM command.
APPEND	Restores all the User-Defined Hierarchies in the procedure set that had been appended with the SAVE CUSTOM <dimension> <setname> APPEND command. Note: You can restore up to the number of User-Defined Hierarchies limitation set in the CUSTOM N statement of the dimension.

Notes:

To restore a User-Defined Hierarchy, you must be either the administrator or the owner.

If you restore the User-Defined Hierarchies for a dimension, it removes any of the dimension's User-Defined Hierarchies that have not been saved.

6.149 RESULT (Dimension)

Use

RESULT is a Dimension statement that defines the top-most consolidated member. It does not roll up into any other members. There can be only one result member.

Syntax

RESULT <name> ['<label>']

Parameter	Description
RESULT <name>	Name of the result member, up to 96 bytes (including periods and underscores). Name must begin with a letter. To include a space in a name, replace the space with an underscore (_), or enclose the name in single quotation marks (' '). Result names tend to have a prefix TOTAL_, for example, TOTAL_PRODUCTS.
'<label>'	Specifies a text string to assign to the result member, enclosed in single quotation marks (' '). The '<label>' option is equivalent to the LABEL keyword in a CONSTRUCT command.

Remarks

Use SET SHORT|LONG to determine whether member names or labels are displayed.

You must not place the rollup rules immediately after the result member if you have not specified a label for the result member. This is because Application Server will be unable to determine whether the first token in the rollup rules is intended for use as a label or as a rollup rule. To avoid this problem, do one of the following:

Specify a label for the result member

Place a CLASS or a LEVEL statement between the RESULT statement and the rollup rules.

Example

```

INPUT
  BOSTON
  ,NEW_YORK
  ,ATLANTA
  ,ORLANDO
  ,HOUSTON
  ,CHICAGO
  ,LOS_ANGELES
  ,SEATTLE
OUTPUT
  NORTHEAST
  ,SOUTHEAST
  ,MIDWEST
  ,WEST
RESULT
  TOTAL_REGION
NORTHEAST= SUM BOSTON, NEW_YORK
SOUTHEAST= SUM ATLANTA, ORLANDO
MIDWEST= SUM HOUSTON, CHICAGO
WEST= SUM LOS_ANGELES, SEATTLE
TOTAL_REGION = SUM NORTHEAST, SOUTHEAST, MIDWEST, WEST

```

6.150 RETURN

Use

RETURN stops the current process.

ROLLUP

Syntax

RETURN

6.151 ROLLUP

Syntax

ROLLUP [<variable> | <dimension1>, <dimension2>, ..., <dimension_n>]

Editor Syntax

```
[ ADD {<number> [..., <number>] | <dimension> {Input|Output} [..., <dimension> {INPUT|OUTPUT}} |
  EVERYBODY } ]

[ REMOVE {<number> [..., <number>] | <dimension> {Input|Output} [..., <dimension> {INPUT|OUTPUT}} |
  EVERYBODY } ]

[ SHOW [COUNT] ]

[ CHANGE
  {<number> [..., <number>] | <dimension> { INPUT|OUTPUT } [..., <dimension> {
INPUT|OUTPUT } } ]
  {
    {UPDATE [OVERWRITE | NOOVERWRITE] [CONSOLIDATE <dimension>
[,...,<dimension>] ] } |
    {DYNAMIC | STATIC} |
    {CONSOLIDATE} |
    {PRECONSOLIDATED | UNCONSOLIDATED} } ]

END | QUIT
```

Use

When you enter ROLLUP <variable>, Application Server creates a table based on the dimensions associated with that variable and all other variables with the same dimensions. You can view the table using the SHOW or SHOW COUNT command in the Rollup editor. You can remove, change, or add quadrants in the table. When you specify a dimension list with the ROLLUP command, all current and future variables dimensioned by the names in the list are affected by the subsequent Rollup editor commands.

The ROLLUP command starts the Rollup editor, which allows you to do any of the following:

Display the quadrants and their rollup instructions and their percentage that they are already consolidated. See the ROLLUP SHOW syntax.

Specify the member combinations (quadrants) to use in a fast or smart consolidation. A smart or fast consolidation follows a strictly additive rule, where it gets each series from the database for a particular variable and adds it to its parent. Only simple additions and subtractions are used to roll up child series into their parents. See the ROLLUP ADD syntax.

Remove time-series combinations that Application Server does not need for reporting and analysis. By specifying the unnecessary combinations that you do not want to be consolidated, it reduces the size of the database and increases calculation speeds. See the ROLLUP REMOVE syntax.

Perform a normal consolidation, which executes the consolidation statements defined in the dimension. You might want to perform a normal consolidation when data is loaded at the output level for certain combinations.

Specify that certain member combinations will be consolidated on the fly on an as needed basis. The combinations would be consolidated whenever any commands are issued that require data

about the combinations. The aggregations are not saved in the database from session to session. See the ROLLUP CHANGE <numbers> DYNAMIC syntax.

Specify that certain data is loaded at the output level for certain combinations See the ROLLUP CHANGE <number> UPDATE CONSOLIDATE syntax.

Note: The number of potential time-series combinations for a variable is calculated by multiplying together the total number of members for each dimension that the variable is dimensioned by. If your dimension includes an ALLOCATE statement you must use the total allocated not the actual number contained in the dimension. The maximum is 18,446,744,060,000,000 (1.8446744 e19) combinations. If you exceed the combination limit, you could redesign the model by either removing the variable and recreating it with fewer dimensions, reducing the number of levels in a dimension, or changing levels into attributes.

Parameter	Description
<variable>	Name of a variable whose dimension combinations you want to include or exclude. Variable names that use special characters should be in single quotation marks (' ').
<dimension1>, <dimension2>, ..., <dimension_n>	A list of dimensions whose combinations you want to include or exclude.

Editor Syntax

ADD {<number> [..., <number>] | <dimension> { INPUT|OUTPUT } [..., <dimension> { INPUT|OUTPUT }} | EVERYBODY }

ADD	Adds the specified combinations.
<number>	One or more numbers, or ranges of numbers, separated by commas. Each number represents a particular combination of input and output members.
<dimension>	Specifies the name of the dimension to be affected by the CHANGE command. Note: The list used to qualify the member combinations can contain all or some of the dimensions for the specified variable. Dimensions not specified in the list automatically have both input and output members included as part of the overall combination definition.
Input	Specifies that the dimension's input level will be affected by the CHANGE command. You can specify the Input keyword or simply the letter I.
Output	Specifies that the dimension's output level will be affected by the CHANGE command. You can specify the Output keyword or simply the letter O.
EVERYBODY	Adds all combinations.

REMOVE {<number> [..., <number>] | <dimension> { INPUT|OUTPUT } [..., <dimension> { INPUT|OUTPUT }} | EVERYBODY }

REMOVE	Removes (does not consolidate) the combinations that are not relevant. When you exclude the combinations for a specific variable, the combinations for all other variables with the same dimension structure are excluded.
<number>	One or more numbers, or ranges of numbers, separated by commas. Each number represents a particular quadrant.
<dimension>	Specifies the name of the dimension to be affected by the CHANGE command. Note: The list used to qualify the member combinations can contain all or some of the dimensions for the specified variable. Dimensions not specified in the list automatically have both input and output members included as part of the overall combination definition.
Input	Specifies that the dimension's input level will be affected by the CHANGE command. You can specify the Input keyword or simply the letter I.
Output	Specifies that the dimension's output level will be affected by the CHANGE command. You can specify the Output keyword or simply the letter O.
EVERYBODY	Removes all combinations (default).

ROLLUP

Note: When you exclude the combinations for a specific variable, the combinations for all other variables with the same dimension structure are excluded.

SHOW [COUNT]

SHOW

Displays information about the quadrants for the specified variable.

COUNT

Displays the percentages and counts of consolidated data and low-level inputs.

The SHOW keyword displays the following symbols to denote the various types:

* means that there is data entered for these input combinations.

+ identifies the dimension to be consolidated.

& means that there's data input at output levels, but existing series will not be overwritten during consolidation. This identifies that a NOOVERWRITE keyword was issue.

! means that the quadrant is marked with the DYNAMIC keyword and will be consolidated on the fly.

\$ means that the quadrant is marked as PRECONSOLIDATED so it will not be consolidated during a fast consolidation.

CHANGE syntax

Use the CHANGE syntax to do either of the following:

Specify that particular quadrants have input data at the output level and that those quadrants will be consolidated from the output level. For example, you might have channel sales transactions without knowing whether the sales are from direct sales or distributor sales.

Note: Typically, a dimension's input level contains the loaded, raw data. During a consolidation, the input level quadrants are consolidated into the output levels.

Specify that particular quadrants will be consolidated on the fly whenever the combinations are needed in the execution of another command. For example, a LIST, DISPLAY, or CALCULATE command that uses the combination would cause the combination to be consolidated on the fly. This temporary consolidation is not saved in the database from session to session.

CHANGE

```
{<number> [,... , <number>] | <dimension> {INPUT|OUTPUT} [,... , <dimension> { INPUT|OUTPUT }]}
{ { UPDATE [OVERWRITE | NOOVERWRITE] [CONSOLIDATE <dimension> [,...,<dimension>] ] } |
{DYNAMIC | STATIC} |
{CONSOLIDATE} |
{PRECONSOLIDATED | UNCONSOLIDATED} }
```

<number> One or more numbers, or ranges of numbers, separated by commas. Each number represents a particular combination of input and output members whose status will be modified by the CHANGE statement.

<dimension> Specifies the name of the dimension to be affected by the CHANGE command.

Note: The list used to qualify the member combinations can contain all or some of the dimensions for the specified variable. Dimensions not specified in the list automatically have both input and output members included as part of the overall combination definition.

INPUT Specifies that the dimension's input level will be affected by the CHANGE command. You can specify the Input keyword or simply the letter I.

OUTPUT Specifies that the dimension's output level will be affected by the CHANGE command. You can specify the Output keyword or simply the letter O.

UPDATE Turns on an UPDATE status for the quadrant, signifying that this is the quadrant into which data is read in for the variable(s).

Note: After an UPDATE statement is issued, Application Server indicates the changed quadrant with an asterisk (*) when you execute a SHOW statement in the Rollup editor.

OVERWRITE	Specifies that you want to overwrite existing consolidated series during the consolidation.
NOOVERWRITE	<p>Specifies that you do not want to overwrite existing consolidated series during the consolidation. This would preserve existing consolidated data even if it was not additive in a pure additive sense.</p> <p>Note: After a NOOVERWRITE statement is issued, Application Server indicates the changed quadrant with an ampersand (&) when you execute a SHOW statement in the Rollup editor.</p>
CONSOLIDATE <dimensions> [...<dimension>]	<p>Specifies that Application Server loads data at the output level for a dimension and that the output level is not the topmost output level in that dimension. Informs Application Server that more consolidation is required.</p> <p><dimension> is one or more dimension names that have data loaded at an intermediate (not highest) output level, implying that more consolidation is necessary. Separate dimension names with commas.</p> <p>When specified with NOOVERWRITE, the CONSOLIDATE keyword will apply the NOOVERWRITE for loaded, read-only series only in output series that have been created by either ACCESS READ or CALCULATE DATA. The CONSOLIDATE keyword will not preserve any output series created by the CALCULATE command without the DATA keyword or series created in a prior consolidation.</p>
DYNAMIC	<p>Specifies that the quadrants will be consolidated on the fly whenever the combinations are needed in the execution of another command. For example, a LIST, DISPLAY, or CALCULATE command that uses the combination would cause the combination to be consolidated on the fly. This temporary consolidation is not saved in the database from session to session.</p> <p>Note: After a DYNAMIC statement is issued, Application Server indicates the changed quadrant with an exclamation point (!) when you issue a SHOW statement.</p>
STATIC	<p>Specifies that the quadrant, which was previously marked as Dynamic and consolidated on the fly, will now be available for the default consolidation method.</p> <p>Note: After changing a combination back to static, you should perform a fast consolidate to preconsolidate that combination. Otherwise, subsequent LISTs or DISPLAYs or a DataView will not display any data for the combination.</p>
CONSOLIDATE	<p>Turns off the UPDATE status for the specified quadrant. Use CONSOLIDATE when the quadrant no longer has input data read in. Data for that quadrant is consolidated in the normal manner.</p> <p>Note: After an UPDATE CONSOLIDATE, Application Server indicates the changed quadrant with an asterisk (*) when you execute a SHOW statement in the Rollup editor.</p>
PRECONSOLIDATED	<p>Specifies that everything in this quadrant is completely consolidated, so this quadrant can be ignored completely during a fast consolidation. No series need be consolidated into this quadrant. During a fast consolidate, this quadrant is skipped, and this speeds up consolidation.</p> <p>Note: After a PRECONSOLIDATED statement is issued, Application Server indicates the changed quadrant with a dollar sign (\$) when you issue a SHOW statement.</p>
UNCONSOLIDATED	Specifies that some series in this quadrant need to be consolidated during a fast consolidation. If the quadrant is also set to OVERWRITE, then existing series will be overwritten.
END QUIT	
END	Exits from the Rollup editor and saves changes.
QUIT	Exits from the Rollup editor without saving changes.

Example

...Assume the Region, Product, and Manufacturer dimensions have the following levels:

ROLLUP

MANUFACTURER	PRODUCT	REGION
TOTAL	TOTAL	Highest level
	LINE	COUNTRY
OWNERSHIP	BRANDS	STATE
NAME	ITEM	City
		Lowest level

...Sales data exists at the Country level in the Region dimension, but is not provided at the lower ... levels. The ROLLUP SHOW statement reveals that quadrant 4 contains loaded data.

ROLLUP Sales

ADD EVERYBODY

SHOW

#	MANUFACTURER	PRODUCT	REGION	%
1	Input	Input	Input	0.0
2	Output	Input	Input	0.0
3	Input	Output	Input	0.0
4	Input	Input	Output	100.0
5	Output	Output	Input	0.0
6	Output	Input	Output	0.0
7	Input	Output	Output	0.0
8	Output	Output	Output	0.0

...CHANGE 4 UPDATE changes the status of quadrant 4 to ensure proper consolidation of the data ... loaded at the output level of Region. Because the data is not loaded at the Result level of Region, ...CONSOLIDATE REGION indicates that further consolidation .is required

CHANGE 4 UPDATE CONSOLIDATE REGION

Changing the Following:

#	MANUFACTURER	PRODUCT	REGION
4*	Input	Input	Output

Asterisk Indicates Data Is Entered For These Input/Output Combinations

1 Combination(s) Changed

SHOW

#	MANUFACTURER	PRODUCT	REGION	%
1	Input	Input	Input	0.0
2	Output+	Input	Input	0.0
3	Input	Output+	Input	0.0
4*	Input	Input	Output+	100.0
5	Output	Output+	Input	0.0
6	Output+	Input	Output	0.0
7	Input	Output+	Output	0.0
8	Output+	Output	Output	0.0

...Assume that data is now provided at the lowest level, City, for the Region dimension. You would

... have to enter another **CHANGE** statement to indicate that Application Server should consolidate
...the variables in the normal manner:

ROLLUP Sales

CHANGE 4 CONSOLIDATE

...Alternatively you could use **REMOVE** and **ADD** commands to set the consolidation back to normal:

ROLLUP Sales

REMOVE EVERYBODY

ADD EVERYBODY

...In the following example, data is entered at the output level for Product and Region, implying that
...quadrant 7 must be changed. The Product dimension has data entered at the Result level. Because
... it is the highest output level, it does not require further consolidation. Region has data entered at
... the State output level and this requires further consolidation. You would enter:

ROLLUP SALES

REMOVE EVERYBODY

ADD EVERYBODY

SHOW

#	MANUFACTURER	PRODUCT	REGION	%
1	Input	Input	Input	0.0
2	Output	Input	Input	0.0
3	Input	Output	Input	0.0
4	Input	Input	Output	0.0
5	Output	Output	Input	0.0
6	Output	Input	Output	0.0
7	Input	Output	Output	100.0
8	Output	Output	Output	0.0

CHANGE 7 UPDATE CONSOLIDATE REGION

Changing the Following:

#	MANUFACTURER	PRODUCT	REGION
7*	Input	Output	Output

Asterisk Indicates Data Is Entered For These Input/Output Combinations

1 Combination(s) Changed

SHOW

#	MANUFACTURER	PRODUCT	REGION	%
1	Input	Input	Input	0.0
2	Output+	Input	Input	0.0
3	Input	Output+	Input	0.0
4	Input	Input	Output+	0.0
5	Output	Output+	Input	0.0
6	Output	Input	Output+	0.0
7*	Input	Output	Output+	100.0

ROUND()

8	Output+	Output	Output	0.0
---	---------	--------	--------	-----

Asterisk Indicates Data Is Entered For These Input/Output Combinations

A Plus Sign(+) Indicates the Dimension to be Consolidated

...In the example above, if data is entered at an output level of the

...Product dimension instead of the Result, the CHANGE command would be:

ROLLUP SALES

CHANGE 7 UPDATE CONSOLIDATE PRODUCT, REGION

6.152 ROUND()

Use

ROUND rounds numbers to the nearest thousand, million, billion, or trillion. You specify the number of decimal places to round.

Syntax

CALCULATE <result> = ROUND(<variable>, <number>)

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable to round. Variable names that use special characters should be in single quotation marks (' ').
<number>	Number of decimal places to round. For example, three rounds up to the nearest thousand and six rounds up to the nearest million.

Example

...This example rounds up the value of Projected_Sales to the nearest thousand:

CALCULATE Projected_Sales = ROUND(Sales, 3)

6.153 ROWS(Report)

Use

ROWS is a Report command that defines the data that appears across report rows. Follow ROWS with a list of expressions, and, optionally, labels associated with each expression. Either the expression or the label prints in the left-most column of that row. ROWS refers to the last down dimension specified, either in a SET-ENDSET block or in an across/down list. The expressions can include variable names.

Syntax

```
ROWS {ALL}
      {[<expression>]}
      {<expression> ['<text>'] [, ..., <expression> ['<text>'] ]}
      {CLASS <name>}
```

Parameter	Description
ALL	All members of the last down dimension. Labels in the left-hand column of the row are member names.
[<expression>]	Member of the last down dimension in position expression. You must use square brackets ([]). On keyboards without square brackets keys, use vertical bars ().

<expression>	Name of a member of the last down dimension or any arithmetic expression involving members.
<text>	Optional label that precedes the specified results of the expression. If you do not supply text, Application Server prints the expression text in the left-most report column. Text can also include dimension names, enclosed in angle brackets (<>).
CLASS	Prints all rows that belong to a specific class.
<name>	The name of the class.

Example: ROWS

...In this example, the last down dimension is Variables. The report prints a row with the label Sales in ... the left-most column followed by the values of the variable Sales, and then another row with the .. label Expenses in the left-most column, followed by the values of the variable Expenses for each ... other dimension combination.

ROWS Sales, Expenses

...In this example, the last down dimension is Variables. The report prints the label Profit in the left- ... most column of each row, followed by the results of the expression Sales - Expenses.

ROWS Sales - Expenses 'Profit'

...In this example, the last down dimension is Product. The report prints the product dimension ... member name (Cars, Trucks, Ambulances) in the left-most column of successive rows, followed by ... the data for that dimension member.

ROWS Cars, Trucks, Ambulances

...In this example, the last down dimension is Time. The report prints the selected ...time periods in the left-most column of successive rows, followed by the data for that time period.

ROWS ALL

...This example prints all the rows in the last down dimension, with a blank line between each row:

SCALAR i

DO i=1, NROWS()

 ROW [i]

 SKIP

ENDDO

...This example would produce totals of MID_PRICE and LUXURY members of the dimension CARS:

SUBTOTAL

 SUBTOTAL

 ROW CLASS MID_PRICE

 ENDSUBTOTAL MID_PRICE_TOTAL

 SUBTOTAL

 ROW CLASS LUXURY

 ENDSUBTOTAL LUXURY_TOTAL

ENDSUBTOTAL GRAND_TOTAL

6.154 SAVE (Access External)

Syntax

SAVE <procedure>

Use

SAVE is an Access External command that saves your current description statements in a procedure, providing you with a default set of descriptions that you can edit to match an external file format.

Parameter	Description
<procedure>	Name of the procedure.

6.155 SAVE CONTROL

Use

SAVE CONTROL creates a procedure with the current values of control variables.

Syntax

SAVE CONTROL <procedure> [SYMBOLIC] [OVERWRITE]

Parameter	Description
<procedure>	Name of the procedure.
SYMBOLIC	Saves the dimension members of a SELECT command by their member names.
OVERWRITE	Overwrites a procedure with the same name.

Example

...This example saves the control variable Logset:

SAVE CONTROL Logset

6.156 SAVE CUSTOM

Use

The SAVE CUSTOM command saves all the User-Defined Hierarchies defined for a dimension in this session. If you do not save custom User-Defined Hierarchies, Application Server deletes them at the end of a session.

Syntax

SAVE CUSTOM <dimension> {<setname> | <owner>.<setname>}

[OMIT <user_defined_hierarchy>]
 [OVERWRITE]
 [PRIVATE | PUBLIC] [CHANGE <user_defined_hierarchy> = { SELECTED | <expression> }]
 [APPEND <user_defined_hierarchy> | DELETE <user_defined_hierarchy>]

Parameter	Description
<dimension>	Name of the dimension or attribute whose User-Defined Hierarchies you want to save.
<setname>	Name of the procedure set in which to save the User-Defined Hierarchy information.
<owner>.<setname>	Specifies the Application Server user who is responsible for the User-Defined Hierarchy and the procedure set in which to save the User-Defined Hierarchy.

	<p>Use SAVE CUSTOM <dimension> <owner>.<setname> to save a User-Defined Hierarchy and its related compiled dimension sets to CGLIB so they can be reproduced again during a RESTORE command.</p> <p><owner> is the Application Server user who created the User-Defined Hierarchy. If you omit the owner name, it is saved to the current user name.</p> <p><setname> is the name of the procedure set in which to save the User-Defined hierarchy.</p> <p>The User-Defined Hierarchy is saved to the current USE database.</p>
OVERWRITE	<p>Overwrites a procedure with the same name.</p> <p>Note: Only the administrator or owner can overwrite the User-Defined Hierarchy.</p>
OMIT <user_defined_hierarchy>	<p>Indicates one User-Defined Hierarchy that will be omitted from the save.</p> <p><user_defined_hierarchy> is the name of the User-Defined Hierarchy to omit.</p>
PRIVATE	<p>Saves the User-Defined Hierarchy in CGLIB with no permissions to any other user but the user who created it. This User-Defined Hierarchy is private and available only to the user who created it.</p>
PUBLIC	<p>Saves the User-Defined Hierarchy in CGLIB with READ permissions for all users. Users can access and use public User-Defined Hierarchies.</p>
CHANGE	<p>Saves any changes to a User-Defined Hierarchy definition that occurred during a CREATE <dimension> REPLACE <user_defined_hierarchy> command, and updates the procedure that defines this User-Defined Hierarchy. This way, all User-Defined Hierarchies that embed this User-Defined Hierarchy in their definition will be updated.</p>
SELECTED	<p>Saves only selected members to this User-Defined Hierarchy.</p>
<expression>	<p><member> ["<label>"] [+ <member> - <member>] [+ <member> - <member>] ... or <user_defined_hierarchy> [+ <user_defined_hierarchy> - <user_defined_hierarchy>] ...</p>
<member>	<p>Members to add or remove from the User-Defined Hierarchy. You can add and subtract members using the plus (+) and minus (-) signs.</p>
<label>	<p>Optional label for a User-Defined Hierarchy member. Must be in double quotation marks (" "). Use up to 250 characters.</p>
<user_defined_hierarchy>	<p>Name of an existing User-Defined Hierarchy to add or remove from the User-Defined Hierarchy. You can add and subtract User-Defined Hierarchies using the plus (+) and minus (-) signs.</p>
APPEND <user_defined_hierarchy> [,<user_defined_hierarchy>]	<p>Adds User-Defined Hierarchies to saved procedure sets.</p> <p><user_defined_hierarchy> is the name of the User-Defined Hierarchy you want to add.</p> <p>For example, in the command SAVE CUSTOM Product prod_udh_set PUBLIC APPEND myproducts, the User-Defined Hierarchy myproducts will be added to the procedure set prod_udh_set for the Product dimension. If the procedure set does not exist yet, it will be created.</p> <p>Note: You can append up to the number of User-Defined Hierarchies that were specified in the CUSTOM <n> statement in the dimension set.</p>
DELETE <user_defined_hierarchy> [,<user_defined_hierarchy>]	<p>Deletes User-Defined Hierarchies from saved procedure sets that were appended using the SAVE CUSTOM APPEND command. <user_defined_hierarchy> is the name of the User-Defined Hierarchy you want to delete.</p> <p>For example, in the command SAVE CUSTOM Product prod_udh_set PUBLIC DELETE myproducts, the User-Defined Hierarchy myproducts will be deleted from the procedure set prod_udh_set for the Product dimension.</p>

Notes:

SAVE DATA

When you use SAVE CUSTOM with PUBLIC or PRIVATE, the User-Defined Hierarchy is saved to the procedure set and the compiled dimension changes are saved too. When you execute a RESTORE command in the next session, the compiled dimension changes are copied from CGLIB and the User-Defined Hierarchies are restored. This avoids having to recreate the User-Defined Hierarchies in every new session, and the User-Defined Hierarchies can be quickly retrieved when you restore them.

When you use SAVE CUSTOM without the PUBLIC or PRIVATE keywords, the User-Defined Hierarchies are saved to the procedure set so they can be recreated for the next session when you execute the procedure set. The User-Defined Hierarchies are available only to the user who created them.

Only the administrator or owner can save a User-Defined Hierarchy using the OVERWRITE keyword.

Example

...This example creates a User-Defined Hierarchy for the Region dimension called My_Reg and then
... saves all the Region dimension's User-Defined Hierarchies to a procedure called My_Reg_grp:

```
CREATE Region My_Reg=MA+NH
```

```
SAVE CUSTOM Region My_Reg_grp
```

...The procedure set would look like this:

```
TYPE MY_REG_GRP
```

```
Begin
```

```
Create REGION MY_REG =
```

```
+MA
```

```
+NH
```

```
End
```

6.157 SAVE DATA

Use

SAVE DATA saves temporary variable data created with CALCULATE.

Use SAVE DATA to save data stored in temporary variables. Application Server saves data stored in permanent variables when you exit from Application Server.

Syntax

SAVE DATA [<variables>]

Parameter	Description
<variables>	One or more temporary variable names separated by commas (.). If you do not specify a variable name, Application Server saves all temporary variables. Variable names that use special characters should be in single quotation marks ('').

Example: SAVE DATA

...This example saves all data:

```
SAVE DATA
```

6.158 SAVE STATUS

Use

Creates a procedure that you can use to recreate the current status settings.

SAVE STATUS also saves the settings of any control variables in effect. To save only the control variables, use SAVE CONTROL.

Syntax

SAVE [<dimension>] STATUS <procedure> [OVERWRITE] [SYMBOLIC] [SCRIPT] [ORDER] [EARLIEST] [LATEST] [USE]

Parameter	Description
<dimension>	Specify this parameter to save just the selection status for the specified dimension.
<procedure>	The name of the procedure you want Application Server to create.
OVERWRITE	Overwrites a procedure with the same name as <procedure>.
SYMBOLIC	Specify the SYMBOLIC keyword to save the selected dimension members using their member names. If you omit this keyword, Application Server saves the selected members using their numeric position, for example, #1, #2, #3, and so on. Note: If you use SYMBOLIC in conjunction with the SCRIPT keyword, numeric member name references in the SELECT command are replaced in the SAVE STATUS procedure with the short member names.
SCRIPT	Saves the last SELECT command, rather than the list of members currently selected. If the dimension(s) changes, this option guarantees the same result, whereas members would be different. Note: If you use SCRIPT in conjunction with the SYMBOLIC keyword, numeric member name references in the SELECT command are replaced in the SAVE STATUS procedure with the short member names.
ORDER	Stores the last ORDER command for each dimension, if any. This option is only effective if the last SELECT command for that dimension was not SELECT <dimension> WHERE.
EARLIEST	Stores the earliest date in the defined procedure set. Normally, this is stored in the set in a comment format — that is, preceded by an ellipsis (...). For example: ... Set Earliest 01 Jan 1999
LATEST	Stores the latest date in the defined procedure set. Normally, this is stored in the set in a comment format — that is, preceded by an ellipsis (...). For example: ... Set Latest 30 Jun 2000
USE	Adds a line to the procedure to use the current Use database.

Example

...This example shows what happens for a given status situation:

STATUS

Across List: # Selected

TIME

Down List:

MERCHANDISE 2 FOOTWEAR, SPORTSWEAR

VARIABLES 1 SALES

SAVE STATUS Setup

TYPE PROCEDURE Setup

SD()

Select sales

Select merchandise footwear, sportswear

Across time Down merchandise, variables

...This example saves just the selection status for a single dimension, FRED:

SAVE PRODUCT STATUS FRED

6.159 SD()

Use

SD returns the standard deviation of the specified variables.

Syntax

CALCULATE <result> = SD(<input> [..., <input>])

Parameter	Description
<result>	Name of the result variable.
<input>	One or more variable names for which you want to calculate the standard deviation.

6.160 SELECT <dimension>

Use

Selects a dimension or a dimension's members for data input, analysis, or output. A SELECT command remains in effect until you enter another statement.

Syntax

```

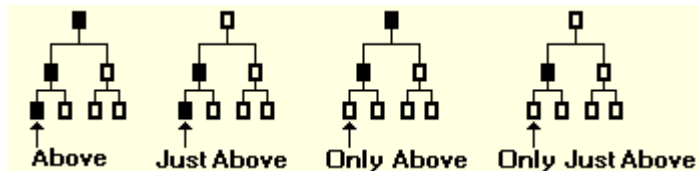
SELECT <dimension>[;<database>] [PLUS | MINUS]
  { <members> }
  { USING <variable> [PERIOD <period>] }
  { [EQ | IS | NE | ISNT] <members> }
  { ABOVE <member> | BELOW <member> }
  { ONLY { ABOVE <member> | BELOW <member> } }
  { JUST { ABOVE <member> | BELOW <member> } }
  { ONLY JUST { ABOVE <member> | BELOW <member> } [SECURITY] }
  { DRILL ABOVE <member> | DRILL BELOW <member> }
  { AFTER <member> | BEFORE <member> }
  { WHERE [<attribute>] [<expression>] [<member> ] [ACROSSDOWN] }
  { INPUT | OUTPUT | RESULT | CUSTOMGROUPS }
  { NONE }
  { <user_defined_hierarchy> }
  { BELOW <user_defined_hierarchy> }
  { LEVEL <level> [ MINUS CUSTOMGROUPS] | [ WHERE [<attribute>] [<expression>] [<member>] ] }

```

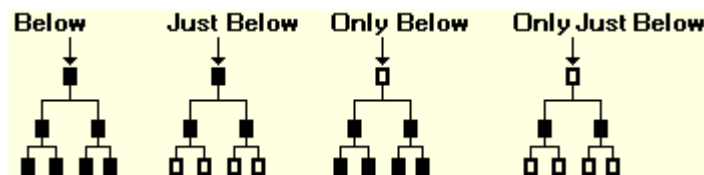
Parameter	Description
<dimension>	Name of the dimension whose members you want to select. To select dimension members from a database other than the Use database, suffix the dimension name with ;<database>, where database is the name of an attached database. For example, SEL Product;Juice
;<database>	By default the SELECT <dimension> command looks for the dimension only in the current model. You can select a dimension from an attached dimensional model. Separate the dimension and database name with a semi-colon (;).
PLUS	Adds the specified members to those already selected.
MINUS	Removes the specified members from those already selected.

Note: If you do not use the PLUS or MINUS keywords, Application Server only selects those members that meet the selection criteria specified in the expression.

<members>	Name of one or more dimension members, levels, classes, or User-Defined Hierarchies, separated by commas, for example, SELECT <dimension> #1,#10. You can also use an asterisk (*) in a name as a wildcard that indicates that any character can occupy that position or any of the remaining positions in the name. You can also specify a range; for example, SELECT <dimension> #1-30 would select the first thirty members.
USING <variable>	Examines all series of the measure and selects those members of the relevant dimension for which the series is not empty. When you are using a Security procedure to limit access, then SELECT dimension USING will only return members that you have access to.
PERIOD <period>	One of the following periods: yty — The latest period back to the equivalent period one year ago. yrago — The latest period one year ago. current — The period set with a SET LATEST command. previous — The previous period. latest — The period set with a SET LATEST command.
EQ IS	Selects all specified members.
NE ISNT	Selects all members other than those specified.
ABOVE	Selects the specified member and all the members in the path above it. If the dimension contains multiple hierarchies, <member> is a member in the currently set hierarchy. SELECT ABOVE selects the specified member and all the members in the same hierarchy above it.
<member>	Name of the member.
JUST ABOVE	Selects the specified member and the member in the path directly above it.
ONLY ABOVE	Selects all members in the path above the specified member, but not including the specified member.
ONLY JUST ABOVE	Selects only the member in the path directly above the specified member.



BELOW	Selects the specified member and all members in the path below it. If the dimension contains multiple hierarchies, <member> is a member in the currently set hierarchy. SELECT BELOW selects the specified member and all the members in the same hierarchy below it.
JUST BELOW	Selects the specified member and the members directly below it.
ONLY BELOW	Selects all members in the path below the specified member, but not including the specified member.
ONLY JUST BELOW	Selects only the members in the path directly below the specified member.



SECURITY	Filters the selection to only those items that have security applied in a Security procedure.
----------	---

SELECT <dimension>

DRILL ABOVE

Selects the members across all paths directly above the specified member, but not including the specified member.

If the dimension contains multiple hierarchies, member is a member in the currently set hierarchy. DRILL ABOVE selects the members above the specified member in the currently set hierarchy, but not including the specified member.

Note: If you select a User-Defined Hierarchy in the last drill down dimension, drill down into that User-Defined Hierarchy to the base member, and then drill back up, all other level members display, not just those in the User-Defined Hierarchy you originally selected. If you then drill up on one of these other members, and then drill down, the User-Defined Hierarchy you originally selected will no longer appear in the down dimension.

DRILL BELOW

Selects the members directly below the specified member regardless of hierarchy, but not including the specified member.

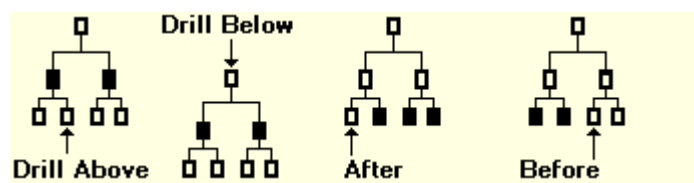
Note: If you select a User-Defined Hierarchy in the last drill down dimension, drill down into that User-Defined Hierarchy to the base member, and then drill back up, all other level members display, not just those in the User-Defined Hierarchy you originally selected. If you then drill up on one of these other members, and then drill down, the User-Defined Hierarchy you originally selected will no longer appear in the down dimension.

AFTER

Selects all input members after the specified member, based on their order in the dimension definition.

BEFORE

Selects all input members before the specified member, based on their order in the dimension definition.

**WHERE <expression>**

Selects all members of the currently selected dimension (and the default member of other dimensions) for which the specified expression is true for any period. The parameter <expression> is a logic expression.

Note: You cannot select dimensions using a WHERE clause. Instead, use BEFORE, AFTER, ABOVE and BELOW, and so on. For example, if you have a dimension called TEST, instead of entering:

SELECT TEST WHERE TEST GE '00:15'

enter:

SELECT TEST AFTER '20:15'

ACROSSDOWN

Use this keyword if you have attributes selected and included in your across/down list, or if the dimension that you are selecting is an attribute dimension. ACROSSDOWN indicates that the data contained in the model's attributes needs to be considered during the SELECT. Otherwise, the attribute's result member will be used, even if the result is not selected. If your dimensional model (or the current view) does not contain attributes, this keyword is ignored.

INPUT

Selects the dimension's input members. If the dimension contains multiple hierarchies, INPUT lists only the input members that are part of the currently set hierarchy.

OUTPUT

Selects the dimension's output members. If a dimension contains multiple hierarchies, OUTPUT lists only the output members that are part of the currently set hierarchy.

RESULT

Selects the dimension's result member.

CUSTOMGROUPS

Selects the dimension's User-Defined Hierarchies.

NONE	Used in an INDEX USER-CASE *-ENDINDEX construct in a procedure as a default CASE to prohibit the view of a dimension from all the users without an assigned view.
<attribute>	Name of the attribute you want to select.
<user_defined_hierarchy>	Name of the User-Defined Hierarchy you want to select based on the CREATE <dimension> <user_defined_hierarchy> command.
BELOW <user_defined_hierarchy>	Used for selecting the members below the User-Defined Hierarchy as well as the User-Defined Hierarchy itself. Use this to show the total User-Defined Hierarchy value and the members that make up the total.
LEVEL	Specifies a dimension level, as in: SELECT Product LEVEL Product_Name where Product_Name is a dimension level of the dimension Product.
<level>	Name or number of the dimension level.
MINUS CUSTOMGROUPS	Selects members at a given level and excludes User-Defined Hierarchies.

6.161 SELECT <attribute>

Use

SELECT <attribute> selects an attribute, an attribute's members, or a dimension associated with an attribute for data input, analysis, or output.

Syntax

```
SELECT <attribute>[:<database>] [PLUS | MINUS]
  { <members> }
  { USING <variable> [PERIOD <period>] }
  { [EQ | IS | NE | ISNT] <members> }
  { [ONLY] ABOVE <member> | BELOW <member> }
  { [JUST] ABOVE <member> | BELOW <member> }
  { [ONLY JUST] ABOVE <member> | BELOW <member> }
  { DRILL ABOVE <member> | DRILL BELOW <member> }
  { AFTER <member> | BEFORE <member> }
  { WHERE [<dimension>] [<expression>] [<member>] [ACROSSDOWN] }
  { INPUT | OUTPUT | RESULT }
  { NONE }
  { <user_defined_hierarchy> }
  { BELOW <user_defined_hierarchy> }
  { LEVEL <attlevelname> WHERE [<dimension>] [<expression>] [<member>] }
```

Parameter	Description
<attribute>	Name of the attribute whose members you want to select. To select attribute members from a database other than the Use database, suffix the attribute name with ;<database>, where database is the name of an attached database. For example, SEL Flavor;Juice.
;<database>	By default the SELECT <attribute> command looks for the attribute only in the current model. You can select an attribute from an attached dimensional model. Separate the attribute and database name with a semi-colon (;).
PLUS	Adds the specified members to those already selected.
MINUS	Removes the specified members from those already selected. Note: If you do not use the PLUS or MINUS keywords, Application Server only selects those members that meet the selection criteria specified in the expression.
<member>	Member of the attribute you want to select. Separate multiple members with commas. You can also use an asterisk (*) in a name as a wildcard that indicates

SELECT <attribute>

that any character can occupy that position or any of the remaining positions in the name. You can also specify a range; for example, SELECT <attribute> #1-30 would select the first thirty members.

USING <variable>

Examines all series of the measure and selects those members of the relevant dimension for which the series is not empty.

PERIOD <period>

One of the following periods:

yty — The latest period back to the equivalent period one year ago.

yrago — The latest period one year ago.

current — The period set with a SET LATEST command.

previous — The previous period.

latest — The period set with a SET LATEST command.

EQ | IS

Selects all specified members.

NE | ISNT

Selects all members other than those specified.

ABOVE

Selects the specified member and all the members in the path above it.

If the attribute contains multiple hierarchies, <member> is a member in the currently set hierarchy. SELECT ABOVE selects the specified member and all the members in the same hierarchy above it.

<member>

Name of the member.

JUST ABOVE

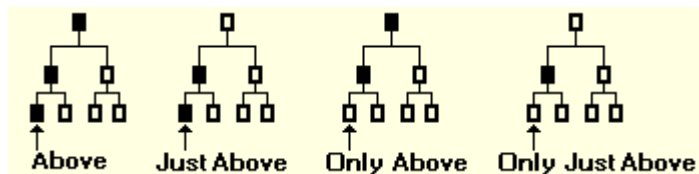
Selects the specified member and the member in the path directly above it.

ONLY ABOVE

Selects all members in the path above the specified member, but not including the specified member.

ONLY JUST ABOVE

Selects only the member in the path directly above the specified member.



BELOW

Selects the specified member and all members in the path below it.

If the attribute contains multiple hierarchies, <member> is a member in the currently set hierarchy. SELECT BELOW selects the specified member and all the members in the same hierarchy below it.

JUST BELOW

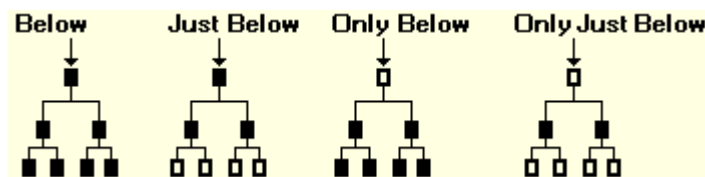
Selects the specified member and the members directly below it.

ONLY BELOW

Selects all members in the path below the specified member, but not including the specified member.

ONLY JUST BELOW

Selects only the members in the path directly below the specified member.



DRILL ABOVE

Selects the members across all paths directly above the specified member, but not including the specified member.

If the attribute contains multiple hierarchies, member is a member in the currently set hierarchy. DRILL ABOVE selects the members above the specified member in the currently set hierarchy, but not including the specified member.

DRILL BELOW

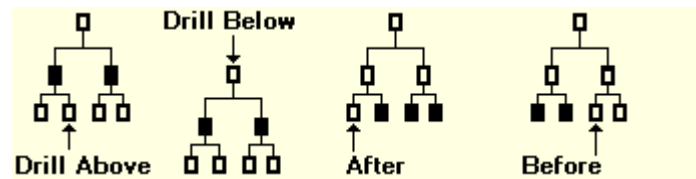
Selects the members directly below the specified member regardless of hierarchy, but not including the specified member.

AFTER

Selects all input members after the specified member, based on their order in the attribute definition.

BEFORE

Selects all input members before the specified member, based on their order in the attribute definition.

**WHERE <expression>**

Selects all members of the currently selected attribute (and the default member of other attribute) for which the specified expression is true for any period. The parameter <expression> is a conditional expression such as IS or GT.

Note: You cannot select attributes using a WHERE clause. Instead, use BEFORE, AFTER, ABOVE and BELOW, and so on. For example, if you have a attribute called Flavor, instead of entering:

SELECT Flavor WHERE Flavor GE '00:15'

enter:

SELECT Flavor AFTER '20:15'

ACROSSDOWN

Use this keyword if you have attributes selected and included in your across/down list, or if the dimension that you are selecting is an attribute dimension. ACROSSDOWN indicates that the data contained in the model's attributes needs to be considered during the SELECT. Otherwise, the attribute's result member will be used, even if the result is not selected. If your dimensional model (or the current view) does not contain attributes, this keyword is ignored.

INPUT

Selects the attribute's input members. If the dimension contains multiple hierarchies, INPUT lists only the input members that are part of the currently set hierarchy.

OUTPUT

Selects the attribute's output members. If a dimension contains multiple hierarchies, OUTPUT lists only the output members that are part of the currently set hierarchy.

RESULT

Selects the attribute's result member.

NONE

Used in an INDEX USER-CASE *-ENDINDEX construct in a procedure as a default CASE to prohibit the view of an attribute from all the users without an assigned view.

<attribute>

Name of the attribute you want to select.

<user_defined_hierarchy>

Name of the User-Defined Hierarchy you want to select based on the CREATE <dimension> <user_defined_hierarchy> command.

BELOW <user_defined_hierarchy>

Used for selecting the members below the User-Defined Hierarchy as well as the User-Defined Hierarchy itself. Use this to show the total User-Defined Hierarchy value and the members that make up the total.

LEVEL

Specifies an attribute level, as in:

SELECT Flavor LEVEL Ades

where Ades is an attribute level of the attribute Flavor.

<attlevelname>

Name of the attribute level.

Selecting an attribute variable and a regular variable

You cannot select an attribute variable and a regular variable with the same SELECT command if you list the attribute variable. For example, if an attribute variable FLAVOR is dimensioned by PRODUCT and a variable SALES is dimensioned by PRODUCT and REGION, enter the command:

SELECT SALES,FLAVOR

do not enter:

SELECT measures

SELECT FLAVOR,SALES

6.162 SELECT measures

Use

SELECT MEASURES selects all non-attribute variables for subsequent operations.

Syntax

SELECT MEASURES

```
{VIRTUAL | NONVIRTUAL}
{ EXCLUDE }
{ [ONLY] [JUST] BELOW <virtual_variable> [VIRTUAL | NONVIRTUAL] }
{ [ PLUS | MINUS ] <variables> [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
{ [ PLUS | MINUS ] [BY <dimensions> | UNDIMENSIONED] [<periodicity>] [VIRTUAL | NONVIRTUAL]
[EXCLUDE] }
{ [ PLUS | MINUS ] FROM {<report> | <logic> | <virtual_variable> } [RECURSE] [TEMPORARY |
PERMANENT] [SORT [REVERSE] ] [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
{ [INPUT | OUTPUT] [<periodicity>] [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
```

Parameter	Description
MEASURES	Selects only non-attribute variables in your Use database. Note: In all cases, if you select measures without a PLUS or MINUS keyword, Application Server always deselects any previously selected variables and then selects according to the new SELECT command.
VIRTUAL	Selects virtual measures. This keyword can be used with other SELECT MEASURE keywords.
NONVIRTUAL	Selects non virtual measures. This keyword can be used with other SELECT MEASURE keywords.
EXCLUDE	Selects the specified variables, but omits hidden variables.
ONLY BELOW	Selects all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also selected. The specified virtual variable is not selected.
JUST BELOW	Selects the specified virtual variable and all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is selected, but the variables that make up that virtual variable are <i>not</i> selected.
BELOW	Selects the specified virtual variable and all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also selected.
ONLY JUST BELOW	Selects all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is selected, but the variables that make up that virtual variable are <i>not</i> selected. The specified virtual variable is not selected.
<virtual_variable>	Specifies the virtual variable whose variables and virtual variable hierarchies you want to select.
PLUS	Selects the specified variables in addition to those currently selected.
MINUS	Removes the specified variables from those currently selected.
<measures>	One or more variable names or labels separated by commas. You can use the following qualifiers: OF, IN, FROM, and VERSION. VERSION can be abbreviated to one character. You can use the qualifier, yrago. You can follow a variable name with an alternate name, enclosed in double quotation marks (" "), to use instead of the default name.

BY <dimensions>	Indicates the variable is dimensioned by the specified dimensions, where <dimensions> is one or more dimension names separated by commas. Note: You must specify all the dimensions the variable is dimensioned by.
UNDIMENSIONED	Selects all undimensioned variables.
<periodicity>	Selects only those variables with one of the following periodicities: constant, yearly, semiannually, quarterly, monthly, lunar, bimonthly, biweekly, weekly, daily, or hourly.
FROM	Selects variables from the specified report or logic set or virtual variable.
<report>	Selects variables from the specified report.
<logic>	Selects variables from the specified logic set.
<virtual_variable>	Selects variables from the specified virtual variable.
RECURSE	Selects the variables referenced in the logic, report, or virtual variable. If a virtual variable is used in the specified logic, report, or virtual variable, then RECURSE also selects the variables that compose that virtual variable.
SORT	Sorts the variables in alphabetical order from a to z.
REVERSE	Reverses the sort order of variables so they are listed from z to a.
TEMPORARY	Selects temporary variables in the specified report or logic set.
PERMANENT	Selects permanent variables in the specified report or logic set.
Note: By default, both TEMPORARY and PERMANENT variables are selected.	
INPUT	Selects input variables in the specified report or logic set.
OUTPUT	Selects output variables in the specified report or logic set.
Note: By default, both INPUT and OUTPUT variables are selected.	

6.163 SELECT <variables>

Use

SELECT <variables> selects all variables for subsequent operations.

Syntax

```
SELECT
  { VARIABLES [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
  { VARIABLE [ONLY] [JUST] BELOW <virtual_variable> [VIRTUAL | NONVIRTUAL] }
  {VIRTUAL | NONVIRTUAL}
  {UNDIMENSIONED}
  { [PLUS | MINUS] <variables> [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
  { [PLUS | MINUS] [BY <dimensions> | UNDIMENSIONED] [<periodicity>] [VIRTUAL | NONVIRTUAL]
  [EXCLUDE] }
  { [PLUS | MINUS] FROM {<report> | <logic> | <virtual_variable> } [RECURSE] [TEMPORARY |
  PERMANENT] [SORT [REVERSE] ] [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
  { [INPUT | OUTPUT] [<periodicity>] [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
  { [<variable> [, <variable>...] [, COUNT OF <variable>] [, DISTINCT COUNT BY <dimension>
  [, <dimension>...] OF <variable>] [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
  { <variable> FROM <database> [VIRTUAL | NONVIRTUAL] [EXCLUDE] }
```

Parameter	Description
VARIABLES	Selects all variables in your Use database. Note: In all cases, if you select variables without a PLUS or MINUS keyword, Application Server always deselects any previously selected variables and then selects according to the new SELECT command.
<variables>	Selects one or more variable names or labels separated by commas. You can use the following qualifiers: OF, IN, FROM, and VERSION.
VIRTUAL	Selects virtual measures. This keyword can be used with other SELECT keywords.

SELECT <variables>

NONVIRTUAL	Selects non virtual measures. This keyword can be used with other SELECT keywords.
EXCLUDE	Selects the specified variables, but omits hidden variables.
ONLY BELOW	Selects all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also selected. The specified virtual variable is not selected.
JUST BELOW	Selects the specified virtual variable and all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is selected, but the variables that make up that virtual variable are <i>not</i> selected.
BELOW	Selects the specified virtual variable and all the variables that make up this virtual variable. If a virtual variable is used in the specified virtual variable, then the variables that make up that virtual variable are also selected.
ONLY JUST BELOW	Selects all the variables that compose the specified virtual variable. If a virtual variable is used in the specified virtual variable, then the virtual variable is selected, but the variables that make up that virtual variable are <i>not</i> selected. The specified virtual variable is not selected.
<virtual_variable>	Specifies the virtual variable whose variables and virtual variable hierarchies you want to display.
PLUS	Selects the specified variables in addition to those currently selected.
MINUS	Removes the specified variables from those currently selected.
<variables>	One or more variable names or labels separated by commas. You can use the following qualifiers: OF, IN, FROM, and VERSION. VERSION can be abbreviated to one character. You can use the qualifier, yrago. You can follow a variable name with an alternate name, enclosed in double quotation marks (" "), to use instead of the default name.
BY <dimensions>	Indicates the variable is dimensioned by the specified dimensions, where <dimensions> is one or more dimension names separated by commas. Note: You must specify all the dimensions the variable is dimensioned by.
UNDIMENSIONED	Selects all undimensioned variables.
<periodicity>	Selects only those variables with one of the following periodicities: constant, yearly, semiannually, quarterly, monthly, lunar, bimonthly, biweekly, weekly, daily, or hourly.
FROM	Selects variables from the specified report or logic set or virtual variable.
<report>	Selects variables from the specified report.
<logic>	Selects variables from the specified logic set.
<virtual_variable>	Selects variables from the specified virtual variable.
RECURSE	Selects the variables referenced in the logic, report, or virtual variable. If a virtual variable is used in the specified logic, report, or virtual variable, then RECURSE also selects the variables that compose that virtual variable.
SORT	Sorts the variables in alphabetical order from a to z.
REVERSE	Reverses the sort order of variables so they are listed from z to a.
TEMPORARY	Selects temporary variables in the specified report or logic set.
PERMANENT	Selects permanent variables in the specified report or logic set. Note: By default, both TEMPORARY and PERMANENT variables are selected.
INPUT	Selects input variables in the specified report or logic set.
OUTPUT	Selects output variables in the specified report or logic set. Note: By default, both INPUT and OUTPUT variables are selected.
COUNT OF <variable>	Provides a count of the number of individual time series that make up a particular view.

Note: This feature is available for simple variables, but not virtual variables.

DISTINCT COUNT BY <dimension> [, <dimension>...] OF <variable>

Provides a distinct count of the number of individual time series for that dimension.

Notes:

- You can perform a DISTINCT COUNT BY only on simple variables, not virtual variables.
- You can create up to 255 different DISTINCT COUNT OF transient measures for any one measure.

VARIABLE <variable> FROM <database>

By default, the SELECT VARIABLE command looks for the variable only in the USE database. You can select a variable from an attached database with the following command:

SELECT VARIABLE <variable> FROM <database>

EXCLUDE

Selects the specified variables, but omits hidden variables.

Example: SELECT

...This example selects the members Footwear and Sportswear from the dimension Merchandise:

SELECT Merchandise Footwear, Sportswear

...This example selects all members of the dimension Merchandise:

SELECT Merchandise

...This example selects all merchandise areas where yearly sales are less than 100,000 in any period:

SELECT Merchandise WHERE Sales LT 100000

...Select all products whose profit margin is greater than 0.4 in any period:

SELECT Merchandise PLUS WHERE Margin GT 0.4

...This example selects all merchandise other than footwear:

SELECT Merchandise NE Footwear

...This example selects the products bought by the customers of

...salesperson JUAN_PEREZ based on Sales. Entering USING Sales

...is more efficient and faster than entering WHERE Sales NE MISSING.

SELECT Customer BELOW Juan_Perez

SELECT Product USING Sales

...This example selects the My_products User-Defined Hierarchy:

SELECT Product My_products

...This example selects the members of the My_products User-Defined Hierarchy,

...as well as the User-Defined Hierarchy itself:

SELECT Product BELOW My_products

...This example selects products colored blue:

SELECT Product WHERE Color IS Blue

...This example selects a dimension from an attached database DEMOMH:

SELECT DIMENSION PRODUCT;DEMOMH

...This example selects members 1 and 10.

SELECT dimension #1,#10

...This example selects the first thirty members.

SELECT <variables>

SELECT dimension #1-30

...This example selects at the Product_Name dimension level

...for a specific manufacturer and product size.

SELECT Product LEVEL Product_Name WHERE Manufacturer IS SQUIBB AND Form IS '50MG','100MG'

Example: SELECT ATTRIBUTE

...This example selects products whose sizes are small and medium:

SELECT Product WHERE Size IS Small, Medium

...This example selects products that are the color blue:

SELECT Product WHERE Color IS Blue

...This example selects products that have sales greater than 1,000:

SELECT Product WHERE Sales GT 1000

Example: SELECT measures

...This example selects all non-attribute variables in your database:

SELECT MEASURES

...This example selects all non-attribute variables minus Sales:

SELECT MEASURES MINUS SALES

...This example selects all non-attribute variables plus Flavor:

SELECT MEASURES PLUS FLAVOR

Examples: SELECT variables

...This example selects all variables in your database:

SELECT VARIABLES

...This example selects the variables Sales and Expenses:

SELECT Sales, Expenses

...This example selects all variables dimensioned by Merchandise and Region:

SELECT BY Merchandise, Region

...This example selects all variables in the logic set Budlogic:

SELECT FROM Budlogic

...This example selects all input variables in the logic set Budlogic:

SELECT FROM Budlogic INPUT

...If some variables are dimensioned by Product and Type, and

...others are dimensioned by Product, Region, and Type, you can enter:

SELECT BY product, region, type

...but you cannot type SELECT BY product.

...This example selects all temporary yearly variables dimensioned by Merchandise:

SELECT BY Merchandise yearly TEMPORARY

...This example selects all variables currently selected, plus the variables Sales and Margin:

SELECT PLUS Sales, Margin

...This example obtains the number of products that were red, white, and blue:

SELECT SALES, COUNT OF SALES

ACROSS TIME DOWN COLOR, VARIABLES**LIST**

...The output of this command would be:

Jan Feb Mar

Red

Sales

Count of Sales

White

Sales

Count of Sales

Blue

Sales

Count of Sales

...This example selects SALES from the attached database DEMOMH:

SELECT SALES FROM DEMOMH

Selecting Attributes - Examples

...This example selects products whose sizes are small and medium:

SELECT Product WHERE Size IS Small, Medium

...This example selects products that are the color blue:

SELECT Product WHERE Color IS Blue

6.164 SELECT VERSION

Use

SELECT VERSION selects a version of the data in the database. All subsequent operations relate to the currently selected version.

Syntax

SELECT VERSION <version>

Parameter	Description
<version>	Name of a version you want to select. The default version name is DEFAULT.

6.165 SET

Use

SET defines the default set for an Application Server session.

A default set is the set automatically accessed when you enter commands but do not specify a set name. For example, assume use the command SET LOGIC CASHFLOW to make CASHFLOW the default logic set. If you enter the command LOGIC, Application Server automatically displays CASHFLOW in the Logic Editor; if you enter the command CALCULATE, Application Server automatically calculates CASHFLOW.

If you do not define a default set with the SET command, the Application Server editors display the last accessed logic set, report, or procedure when you do not specify one.

SET BUFFERS

Syntax

```
SET{LOGIC <setname>}
    {PROCEDURE <setname>}
    {REPORT <setname>}
    {SYNONYM <setname>}
    {TIME <setname>}
    {DIMENSION <setname> [HIERARCHY
        {<hierarchy> | DEFAULT} ]}
```

Parameter	Description
<setname>	Name of the default set.
<hierarchy>	<p>Name of a hierarchy to set for the dimension. Application Server uses the currently set hierarchy instead of the default hierarchy in these situations:</p> <p>When you use the SELECT command with ABOVE, BELOW, DRILL ABOVE, or DRILL BELOW, Application Server bases the selection on the currently set hierarchy.</p> <p>When you use the EXHIBIT command with a HIERARCHY statement, Application Server displays information for the dimension's currently set hierarchy.</p> <p>Notes:</p> <ul style="list-style-type: none"> The hierarchy defined in the first HIERARCHY statement of the CONSTRUCT command is the default hierarchy. When you set a hierarchy, its dimension does not automatically become the default dimension unless you enter a separate SET DIMENSION command.
DEFAULT	Unsets the dimension's hierarchy. The default hierarchy is in effect.

Example

...This example defines the logic set CALC_ALL as the default and accesses it:

```
SET LOGIC calc_all
```

```
LOGIC
```

...This example defines Regions as the set hierarchy in the STORE dimension:

```
SET DIMENSION Store HIERARCHY Regions
```

6.166 SET BUFFERS

Use

SET BUFFERS sets the number of buffers for Application Server to use. The optimal number of buffers depends on your system.

Syntax

```
SET BUFFERS <integer>
```

Parameter	Description
<integer>	Number of buffers between 50 and 64,000.

Note: If you set buffers too low or too high, you can slow Application Server response time.

Example

...This example sets the number of buffers to 1000:

```
SET BUFFERS 1000
```


6.167 SET CELLS

Use

SET CELLS defines the maximum number of cells (data points) displayed in a report. This function keeps the size of a page within reasonable limits, which you could easily exceed with an inappropriate across/down specification.

Syntax

SET CELLS <integer>

Parameter	Description
<integer>	Number of cells to display. The default is no limit.

6.168 SET CHARSET

Use

SET CHARSET sets the charset of the input or output.

Syntax

SET CHARSET
 { INPUT | OUTPUT | EXTIN | EXTOUT }
 { ASCII | SHIFTJIS | JEUC | SCEUC | KEUC | UTF8 [BOM {ON | OFF}] | SESSION | ISO2022 |
 ISO88591 | ISO88592 | ISO88593 | ISO88594 | ISO88595 | ISO88596 | ISO88597 | ISO88598 |
 ISO88599 | WINDOWS1250 | WINDOWS1251 | WINDOWS1252 | WINDOWS1253 | WINDOWS1254 |
 WINDOWS1255 | WINDOWS1256 | WINDOWS1257 }

Parameter	Description
INPUT	Identifies the CHARSET of the input.
OUTPUT	Identifies the CHARSET of the output to IDQL or the output sent to Interactive Publisher.
EXTIN	Identifies the CHARSET of the input from an external file.
EXTOUT	Identifies the CHARSET of the output to an external file.
ASCII	Format of charset.
SHIFTJIS	Format of charset.
JEUC	Format of charset.
SCEUC	Format of charset.
KEUC	Format of charset.
UTF8	Sets a UTF8 character set.
BOM ON	Adds a byte order mark (BOM) at the beginning of UTF8 output files when using the EXTOUT UTF8 BOM ON.
BOM OFF	Turns off the display of the byte order mark at the beginning of UTF8 output files when using EXTOUT UTF8 BOM OFF. The default setting is off and the byte order mark is not displayed.
SESSION	Rolls back the CHARSET setting changed during the session.
ISO<num>	Sets an ISO character set. You cannot set this value on an existing database.
WINDOWS<num>	Sets a Windows character set. You cannot set this value on an existing database.

Example 1

SET CHARSET INPUT UTF8

Tells Application Server that the input is encoded by UTF8.

SET CONTROL

SET CHARSET OUTPUT SCEUC

Tells Application Server that the output should be encoded by SCEUC

SET CHARSET EXTIN SHIFTJIS

Tells Application Server that the input from an external file is encoded by SHIFTJIS

SET CHARSET EXTOUT KEUC

Tells Application Server that the output to an external file should be encoded by KEUC

SET CHARSET EXTOUT UTF8 BOM ON

Tells Application Server that the output from an external file is encoded by UTF8 and contains a byte order mark at the beginning of the file.

Example 2

At the beginning, CHARSET OUTPUT is UTF8 and CHARSET SESSION is UTF8.

Send a command:

SET CHARSET OUTPUT KEUC (charset output changes to KEUC)

SET CHARSET OUTPUT SESSION (charset output rolls back to UTF8)

6.169 SET CONTROL

Use

SET CONTROL sets the value of a control variable.

Syntax

SET CONTROL [\$]<variable> { [<value>] | CLOCK | ELAPSED | MILLISEC | DELTA }

Parameter	Description
<variable>	Name of the control variable.
\$	Specifies that if this control variable is undefined when it is executed, treat it as null rather than producing an error. Note: By default, if you have a procedure that executes a standard SET CONTROL for a variable that does not yet exist, you will get an error message. If you need to run a procedure with undefined control variables, but you want the undefined control variables treated as null rather than as error-producing, you must add a dollar sign (\$) as the first character of those control variable name. If Application Server detects an undefined control variable that begins with a \$, that variable will be treated as null and will not cause an error. If Application Server detects an undefined control variable that begins with any character other than \$, that will cause an error as appropriate.
<value>	Value of the control variable.
CLOCK	Stores the current date and time in the control variable.
ELAPSED	Stores the elapsed time (in seconds) since the last SET CONTROL CLOCK or SET CONTROL ELAPSED command.
MILLISEC	Stores the elapsed time (in milliseconds) since midnight (00:00:00), January 1, 1970.
DELTA	Stores the elapsed time in milliseconds since the last SET CONTROL MILLISEC or SET CONTROL DELTA command.

Example

...This example sets and uses the control variable Logset:

SET CONTROL Logset Mylogic**LOGIC &Logset**

...This example assigns a date range to a control variable

...Week1, and then uses it to modify a listing:

SET CONTROL Week1 'Jan 1 1999 – Jan 7 1999'

LIST PERIOD &Week1

6.170 SET DATE

Use

SET DATE defines a date format so that Application Server can resolve ambiguous dates.

Syntax

SET DATE <format>

Parameter	Description
<format>	A combination of Y (year), M (month), or D (day). The default format is YMD.

Example

...The date 7 August 2006 is ambiguous when represented as two-digit numbers separated by slashes:

... 07/08/06. This statement resolves the ambiguity:

SET DATE dmy

...The date 02/03/00 can be interpreted as February 3, 2000, or as 2 March 2000. The following

... statement ensures that it is interpreted as February 3, 2000:

SET DATE mdy

6.171 SET DEFAULT

Use

SET DEFAULT changes the default formats for *all* variables. You can specify multiple options on the same line, but you must not separate them with commas. You can display the current variable formats with the SHOW FORMAT command.

Syntax

```
SET DEFAULT
  [ AFTER <c> | OFF ]
  [ AUTO [OFF] ]
  [ BEFORE <c> | OFF ]
  [ COMMA <c> | , | . | OFF | LOCALE ]
  [ CURRENCY <c> | $ | OFF ]
  [ DECIMALS <n> | LOCALE ]
  [ FIX CURRENCY | NEGATIVE | DECIMAL ]
  [ FLOAT CURRENCY | NEGATIVE | DECIMAL ]
  [ MISSING <string> | BLANK | DEFAULT ]
  [ NEGATIVE <prefix> [ <suffix> ] | DEFAULT ]
  [ PAD [OFF] ]
  [ POINT <c> | . | , | LOCALE ]
  [ REVERSE [OFF] ]
  [ SCALE <n> | OFF ]
  [ WIDTH <n> ]
  [ ZERO <string> | BLANK | DEFAULT | OFF ]
```

SET DEFAULT

Parameter	Description
AFTER '<c>'	A string of up to three characters to appear after values.
AFTER OFF	Removes any previously set string of characters.
AUTO	Performs automatic scaling.
AUTO OFF	Turns off automatic scaling.
BEFORE '<c>'	A string of up to three characters to appear before values.
BEFORE OFF	Removes any previously set string of characters.
COMMA '<c>'	A character to insert as the thousands separator.
COMMA ,	Inserts a comma (,) as the thousands separator.
COMMA .	Inserts a period (.) as the thousands separator.
COMMA OFF	Removes any previously set thousands separator.
COMMA LOCALE	Sets the thousands separator to the character defined in the locale settings in the Windows Control Panel.
CURRENCY '<c>'	A string of up to four characters to insert as the currency symbol.
CURRENCY \$	Inserts a dollar sign (\$) as the currency symbol.
CURRENCY OFF	Removes any previously set currency symbol.
DECIMALS <n>	The number of decimal places to appear in a value. For example, DECIMALS 2 divides the number by ten to the second power, rounding down halves.
DECIMALS LOCALE	Specifies that Application Server should display values with the number of decimal places specified in the Regional settings of the Windows Control Panel.
FIX CURRENCY	Places the currency symbol in the leftmost position.
FIX NEGATIVE	Places the negative sign in the leftmost position.
FIX DECIMAL	Places the decimal point symbol in the leftmost position.
FLOAT CURRENCY	Places the currency symbol adjacent to the value.
FLOAT NEGATIVE	Places the negative sign adjacent to the value.
FLOAT DECIMAL	Places the decimal point symbol adjacent to the value.
MISSING '<string>'	A string of up to seven characters to insert for missing values.
MISSING BLANK	Inserts a blank for missing values.
MISSING DEFAULT	Inserts a dash (-) for missing values.
NEGATIVE '<prefix> [<suffix>]'	<p>A character to appear before or after a negative number. Enclose the character(s) in single quotation marks (' '). To specify a suffix without a prefix, use a space, and then the character you want to appear as a suffix. Enclose the space and the character in single quotation marks (' ').</p> <p>An example of prefix only: NEGATIVE '(' displays negative numbers as (2,123 instead of as -2,123.</p> <p>An example of prefix and suffix: NEGATIVE '()' displays negative numbers as (2,123) instead of as -2,123.</p> <p>An example of suffix only: NEGATIVE ')' displays negative numbers as 2,123) instead of as -2,123.</p>
NEGATIVE DEFAULT	Inserts a negative sign (-) before a negative number.
PAD	Pads numbers with leading zeroes.
PAD OFF	Removes any padding.
POINT '<c>'	A character to use for the decimal point.
POINT .	Uses a period as the decimal point.
POINT ,	Uses a comma as the decimal point.
POINT LOCALE	Sets the decimal point to the character defined in the locale settings in the Windows Control Panel.

REVERSE	Reverses the position of currency and negative symbols. The default is the currency symbol followed by the negative symbol, for example, \$-2.00.
REVERSE OFF	Returns the currency and negative symbols to the previous positions.
SCALE <n>	Scaling factor; divides the number to be displayed by 10 to the power <n>. For example, scale 2 divides the number by ten to the second power, rounding down halves. That means that when you use scale 2, 2.505 is rounded to 2 decimal places to 2.50, not 2.51.
SCALE OFF	Removes any previously set scaling factor.
WIDTH <n>	Number specifying the field width. The maximum width that you can set for a text variable is 50. Note: A text variable's value can contain up to 255 characters. If a text variable's value contains more characters than the width specified in WIDTH <n>, Application Server truncates the variable's values to a width of <n>.
ZERO '<string>'	A string of up to seven characters to use for true zero values.
ZERO BLANK	Inserts a blank for true zero values.
ZERO DEFAULT	Inserts a dash (-) for true zero values.
ZERO OFF	Removes any previous setting and displays a 0 for true zero values.

Remarks

SET DEFAULT specifies the default global formatting settings for all variables. However, if you use SET VARIABLES to change a formatting option for a variable, that variable will no longer use the default global formatting settings specified by SET DEFAULT.

To compare the default global formatting settings with the current formatting settings for a variable, use the SHOW DEFAULT and EXHIBIT VARIABLE varname commands.

If you use SET VARIABLES to change the formatting for a variable, you must use the REMOVEFORMATvariable command if you want to reset the variable so that it uses the global formatting settings.

Example: SET DEFAULT

...This example assigns a width of eight characters without any decimal places to all variables except ...Pct_Change, which is assigned a width of six characters and two decimal places:

```
SET DEFAULT WIDTH 8 DECIMALS 0
```

```
SET VARIABLES Pct_Change WIDTH 6 DECIMALS 2
```

...This example sets the dollar sign (\$) prefix as the default currency for all variables. Then it specifies ... that just the Units variable does not use the prefix.

```
SET DEFAULT CURRENCY $
```

```
SET VARIABLES Units CURRENCY OFF
```

...This example sets two decimal places for the Units variable and three decimal places for all other ... variables. It also assigns the dollar sign prefix (\$) as the default currency all other variables.

...The Units variable is not affected by the default formats.

```
SET VARIABLES Units DECIMAL 2
```

```
SET DEFAULT DECIMALS 3
```

```
SET DEFAULT CURRENCY $
```

6.172 SET DEFAULT Periodicity

Use

SET DEFAULT <periodicity> defines the display format to use for the specified periodicity.

SET DEFAULT Periodicity

Syntax

SET DEFAULT <periodicity> ['<dateformat>']

Parameter	Description
<periodicity>	Any one of the valid periodicities included in the list below. For example, bimonthly, quarterly, and so on.
['<dateformat>']	Any reserved letters, and any text you want to appear in the date format. The reserved letters are preceded by a percent sign (%), and are as follows: y (year), f (fiscal year), m (month), d (day), h (hour, 12-hour clock), t (hour, 24-hour clock), s (am or pm), S (AM or PM), mi (minute), n (period number), and w (weekday name). The lowercase letters y, f, m, d, h, t, mi, and w indicate the starting year, month, day, and so on. The uppercase letters Y, F, M, D, H, T, MI, and W indicate the ending year, month, day, and so on.

Periodicity	Default output	Default date format
avgytd	2000 Avg	%Y Avg
bimonthly	Jan-Feb 00	%3m-%3M %2y
biweekly	01 Jan-14 Jan 00	%2d %3m-%2D %3M %2Y
daily	01 Jan 00	%2d %3m %2y
hourly	12am 30 Jan	%h%s %d %m
lunar	28 Jan 00	%2D %3M %2Y
monthly	Jan 00	%3m %2y
mtd	Jan 00 Mtd	%3M %2Y Mtd
mytd	Jan 00 Ytd	%3M %2Y Ytd
qtd	Jan-Mar Qtd	%3m-%3M Qtd
quarterly	Jan-Mar 00	%3m-%3M %2Y
rmonthly	31/01/00	%D/%2MM/%2Y
rquarterly	Jan-Mar 00	%3m-%3M %2Y
ryearly	1999	%Y
semiannual	Jan-Jun 00	%3m-%3M %2Y
weekly	07 Jan 00	%2D %3M %2Y
wtd	07 Jan 00	%2D %3M %2Y
yearly	2000	%Y
ytd	2000 Ytd	%Y Ytd

Tips:

To display the month in alphabetic character format, use a single M. For example, %2D %3M %2Y gives 31 Jan 00.

To display the month in numeric format, use MM. For example, %2D %2MM %2Y gives 31 01 00.

To specify a single digit day or month, use %D or %MM without a number preceding the letters. For example, %D-%3M-%2Y gives 1-Jan-00.

To display a 2-digit year, use %2Y. For example, %D/%2MM/%2Y gives 31/01/00.

To display a 4-digit year, use %Y without a number preceding the Y. For example, %Y gives 2000.

To display the full weekday name, use %W.

To abbreviate the weekday name, use a number with W. For example, %3w %2d %3m %2y gives Tue 01 Nov 00.

Restoring the default format

To restore the default format for any periodicity, enter the following:

```
SET DEFAULT WEEKLY "
```

where " is two single quotation marks (').

Example: SET DEFAULT periodicity

... This example adds the weekday name for the start day to the default display for the weekly
... periodicity. It also changes the default display for quarterly to use the ending year of the
... fiscal year, as opposed to the calendar year that ends the quarter. This is useful when your fiscal
... year begins in a month other than January.

```
set default weekly '%2d %3m %2y %w'
```

```
set default quarterly '%m-%M %F'
```

6.173 SET DENSE

Use

SET DENSE changes variables that are created with the CREATE variable DENSE keyword.

If this command is in effect before issuing a CREATE variable DENSE command, the DENSE keyword in the CREATE variable command will be ignored and the ON or OFF setting of the SET DENSE command will be used.

If this command is in effect after a database has been dumped, any variable stored with the DENSE keyword will honor the setting of the SET DENSE command during the LOAD.

Once a SET DENSE command has affected the CREATE variable command, the setting of the SET SPARSE command takes affect.

Note: SPARSE is the better method for creating measures when databases are removed, recreated and populated from scratch each time and no additional data for extra periods is added to the series in that database. DENSE may be a better method for creating measures when additional data for extra periods is added because DENSE measures honor the MULTIPLES keyword, while SPARSE ones do not.

Syntax

```
SET DENSE {ON | OFF}
```

Parameter	Description
ON	Confirms the CREATE variable DENSE command and creates the variable as dense.
OFF	Ignores the DENSE keyword on any proceeding CREATE variable DENSE commands.

6.174 SET DETAIL

Use

SET DETAIL provides detail information about a member during a LIST [TABS] or DISPLAY [TABS] command. The output is always in the order DIMENSION, LONG, SHORT, TYPE, LEVEL, PARENTS, CHILDREN. By default, the items are separated by a horizontal bar (|), and excluding any omitted components.

SET DETAIL

Syntax

SET DETAIL

```
{ ALL | NONE | [ACROSS], [DIMENSION], [LONG], [CHILDREN], [DOWN], [LEVEL], [PARENTS],
[ROWS], [SHORT], [SECURITY], [TYPE] }
```

```
[DELIMITER { X'<nn>' | '<printable char>' | '\<escaped char>' ]
```

Parameter	Description
ALL	Displays all detail information for the selections. Using ALL is the same as specifying all these options: SECURITY, ACROSS, DIMENSION, LONG, DOWN, CHILDREN, PARENTS, LEVEL, SHORT, ROWS, TYPE.
NONE	Turns off all options.
DELIMITER	Specifies a delimiter other than a vertical bar () to display between values. If you do not use the DELIMITER keyword, the vertical bar will be used.
X'<nn>'	Any hex number in the form <nn> where <nn> is {0,1,2,3,4,5,6,7,8,9,0,A,B,C,D,E,F}. For example, SET DETAIL ALL DELIMITER X'09' ... hex 09 is horizontal tab
'<printable char>'	Any printable character. For example, SET DETAIL ALL DELIMITER '-'
'\<escaped char>'	Any one of these escape characters: \f Formfeed \n New line \r Carriage return \t Horizontal tab \v Vertical tab \' Single quotation mark \" Double quotation mark \\ Backslash
ACROSS	Limits the detail information to the currently selected across dimensions.
DIMENSION	Displays the member's dimension.
CHILDREN	Displays the number of immediate children for each member.
DOWN	Limits the detail information to the currently selected down dimensions.
LEVEL	Displays a number that represents the level in which this member exists. 1 represents the input level, 2, represents the first output level, 3 represents the next output level, and so on.
LONG	Displays the long name (member label) of the member.
PARENT	Displays a 0 or 1 that represents whether this member is an orphan or has a parent. 0 means that the member is an orphan and 1 means that it has parents.
ROWS	Shows data for the last down dimension members.
SHORT	Displays the short name (member name) of the member.
SECURITY	Reflects the security definitions of the dimension when displaying the output rather than that of the base dimension. For example, if a user has access to the lowest level of a dimension, the SET DETAIL command used with the SECURITY and PARENTS keyword would show that the member has 0 parents. If you don't specify SECURITY in the SET DETAIL keyword list or if you don't use the ALL keyword, then the results do not reflect security definitions.
TYPE	Displays a 0, 1, 2, 3, 4, or 5 that represents whether the member is an input member (0), output member (1), result member (2), User-Defined Hierarchy (3), class (4), or a Concentrate member (5). Note: A Concentrate member is the "All Others" member in applications such as Ranking.

6.175 SET-ENDSET (Report)

Use

SET-ENDSET is a Report command that defines the global parameters you need to print a report.

Syntax

You can specify one or more of the following options on the same line. Do not separate the options with commas.

```
SET
  [ACROSS <across>]
  [AFTER '<c>' | OFF ]
  [ALWAYS [NLEVEL <n>] <member> [, <member>...]]
  [AUTO [ OFF ] ]
  [COLUMNS <n>]
  [COLUMNS ONCE ]
  [COMMA '<c>' | , | . | OFF]
  [CURRENCY '<c>' | , | . | OFF]
  [DECIMALS <list>]
  [DOWN <down>]
  [FIX CURRENCY | NEGATIVE | DECIMAL]
  [FLOAT CURRENCY | NEGATIVE | DECIMAL]
  [INDENT <n>[<n>,...<n>] ["<char>"]]
  [MISSING '<string>' | BLANK | DEFAULT]
  [NAMES <width>]
  [NEGATIVE '<prefix>' [ <suffix> ] ' | DEFAULT]
  [ORDER <columns> | COLTOTAL ['<colheading>']]
  [PAD [OFF]]
  [POINT '<c>' | . | , ]
  [PREFACE <inputid> <resultid> <outputid> | RANK [NLEVEL <n>] [<column>]]
  [PREFIX [ALWAYS <nn>]
  [REVERSE [ OFF ] ]
  [SCALE <n> | OFF]
  [SORT [NLEVEL <n> ] [COLTOTAL {<n> | LAST}] [ASCENDING | BOTTOM | LAST | TOP <n> | VALUE
<n>] <column>]
  [STACK]
  [STYLE {LEVELTOTAL|TABLE} ]
  [SUPPRESS [WHERE <expression>] MISSING | ZERO | BOTH | FORMFEED | COLHEADING |
PARHEADING ]
  [TABS]
  [TABS SINGLE]
```

SET-ENDSET (Report)

[TRUNCATE]

[UNDERLINE <list>]

[WIDTHS <list>]

[ZERO '<string>' | - BLANK | DEFAULT | OFF]

ENDSET

Parameter	Description
ACROSS <across>	Indicates the dimensions that appear across a report page, where across is one or more dimension names separated by commas (,).
AFTER '<c>'	A string of up to three characters to appear after values.
AFTER OFF	Removes any previously set string of characters.
[ALWAYS [NLEVEL <n>] <member> [, <member>...]	For a specific level, always shows the items defined, irrespective of any cut-off criteria. By default, the definitions are applied to the last down dimension rows. You can use the short or long label as the member name.
AUTO	Performs automatic scaling.
AUTO OFF	Turns off automatic scaling.
COLUMNS <n>	Indicates the maximum number of columns on a report page.
COLUMNS ONCE	Displays column headers once per page, regardless of the number of pages that occur. If you do not use this option, Application Server displays the column headers before each PARHEADING.
COMMA '<c>'	A character to insert as the thousands, millions, billions divider.
COMMA ,	Inserts a comma (,) as the thousands, millions, billions divider.
COMMA .	Inserts a period (.) as the thousands, millions, billions divider.
COMMA OFF	Removes any previously set thousands, millions, billions divider.
CURRENCY '<c>'	A string of up to four characters to insert as the currency symbol.
CURRENCY \$	Inserts a dollar sign (\$) as the currency symbol.
CURRENCY OFF	Removes any previously set currency symbol.
DECIMALS <list>	Indicates the number of decimal places used in numeric columns, where <list> is one or more integers separated by commas. An integer and an asterisk (*) before the decimal indicate to repeat the decimal that number of times. For example, 2*2 means two columns that have two decimal places. You can enclose groups of decimals that repeat (for example, 2, 0, 2, 0, ...) in parentheses and preceded by a repetition count, for example, 5(2, 0). Note: On display, numbers are rounded, rather than truncated. DECIMALS rounds down halves, so when you use DECIMALS 2, the value 2.505 is rounded to two decimal places, giving 2.50, not 2.51.
DOWN <down>	Indicates the dimensions that appear down a report page, where <down> is one or more dimension names separated by commas.
FIX CURRENCY	Places the currency symbol in the leftmost position.
FIX NEGATIVE	Places the negative sign in the leftmost position.
FIX DECIMAL	Places the decimal point symbol in the leftmost position.
FLOAT CURRENCY	Places the currency symbol adjacent to the value.
FLOAT NEGATIVE	Places the negative sign adjacent to the value.
FLOAT DECIMAL	Places the decimal point symbol adjacent to the value.
INDENT <n>[<n>,...<n>]["<char>"]	Indicates the number of characters to indent for each down dimension specified, and, optionally, a character "<char>" (for example, ">"), to use for the indenting. The last indent is used for any subsequent dimension in the report.
MISSING '<string>'	A string of characters to insert for missing values. BLANK — Inserts a blank for missing values.

	DEFAULT — Inserts a dash (-) for missing values.
NAMES <width>	Makes row names the specified number of characters wide, where <width> is the number of characters.
NEGATIVE	<p>You can use the following options:</p> <p><prefix> — A character to appear before a negative number. If you specify a prefix without a suffix, enclose it in single quotation marks (' '). For example, NEGATIVE '('.</p> <p><suffix> — A character to appear after a negative number. Enclose the suffix in a set of single quotation marks (' '). For example, NEGATIVE '()' displays negative numbers as (2,123) instead of as -2,123.</p> <p>DEFAULT — Inserts a negative sign (-) before a negative number.</p>
[NEVER [NLEVEL <n>] <member> [<member>...] WHERE <attributename> IS ISNT EQ NE <membername>	<p>For a specified level, never shows the items defined, irrespective of any cut-off criteria. By default, the definitions are applied to the last down dimension rows. You can use the short or long label as the member name.</p>
ORDER <columns>	Displays columns across a page in the specified order, where <columns> is one or more column numbers or expressions, separated by commas. See the REPORT ORDER command.
COLTOTAL	Displays column subtotals with the default 'Total' heading or a specified heading.
'<colheading>'	Specifies the column heading to use, enclosed in single quotes. If you do not specify a column heading, then 'Total' is used by default.
PAD	Pads numbers with leading zeroes.
OFF	Removes any padding.
POINT '<c>'	A character to use for the decimal point.
POINT .	Uses a period as the decimal point.
POINT ,	Uses a comma as the decimal point.
PREFACE	<p>You can use the following options:</p> <p>{'<inputid> <resultid> <outputid>'} — Adds a number, letter, or character next to each report row in column 1, where <inputid> is the identifier for the input level, resultid is the identifier for the result level, and <outputid> is the identifier for the output level. The number identifies the dimension level that this member is a part of.</p> <p>For example, PREFACE '132' places a 1 next to an input member, a 2 next to an output member, and a 3 next to a result member. PREFACE 'abc' places letters next to the rows.</p> <p>RANK [NLEVEL <n>] [<column>] — Specifies the nested level to use. The optional parameter <column> specifies the column to rank on. If you do not specify a value for <column>, the SORT column is used by default.</p> <p>You can use the RANK keyword to display the sorted position within the level, in either the SORT column or any other non-calculated column. The PREFACE RANK flag is followed by a single space.</p>
PREFIX	ALWAYS <nn> — Prefixes items displayed as a result of an ALWAYS specification.
REVERSE	Reverses the position of currency and negative symbols. The default is the currency symbol followed by the negative symbol, for example, \$-2.00.
OFF	Returns the currency and negative symbols to the previous positions.
SCALE <n>	Scaling factor; divides the number to be displayed by 10 to the power <n>. For example, scale 2 divides the number by ten to the second power, rounding down halves. When you use scale 2, the value 2.505 is rounded to two decimal places, giving 2.50, not 2.51.
SCALE OFF	Removes any previously set scaling factor.

SET-ENDSET (Report)

SORT	Sorts the rows of a report, based on the values in a column, in descending order. Can be used with the following options:
COLTOTAL {<n> LAST }	– Sorts the rows of a report by the <n>th COLTOTAL column or by the last column total.
	ASCENDING — Sorts the rows in ascending order.
	BOTTOM <n> — Bottom <n> cut-off criteria.
	LAST — Sorts on the last column in the report.
	NLEVEL <n> — Sorts the rows within a specific nested level. To sort all levels, include a SORT NLEVEL <n> line for each level.
	TOP <n> — Top <n> cut-off criteria.
	VALUE <n> — Cumulative value cut-off (top or bottom, depending on the specified sort order).
	<column> — One or more column numbers separated by commas, indicating the column to sort.
	Notes:
	NA values appear by default at the bottom of report output, and at the top of report output when ASCENDING is used.
	If you calculate a column based on a row position using ROW[<n>] after executing a SORT command, ROW[<n>] always refers to the <n>th sorted row. To refer to the <n>th unsorted row, use the syntax ROW[-<n>] instead.
STACK	Displays labels for each column, corresponding to the Across dimension.
STYLE	Specifies the style of columns and rows.
LEVELTOTAL	Specifies that a rolled up total should be generated for all down dimensions except the last. Also specifies the execution of the PARHEADING-ENDPARHEADING and SUMMARY-ENDSUMMARY blocks for every level change, allowing the display of total rows for each level.
TABLE	Displays the down dimensions in separate columns with the member names repeated on every row.
SUPPRESS	<p>Suppresses form feeds. It can also suppress rows based on missing values or zero values. The options are:</p> <p>WHERE <expression> — selectively suppresses or displays rows. The parameter <expression> is a logic expression. The example below will suppress all rows that have no value in columns 1, 2, and 3. The operators AND and OR can be used along with EQ, LT, GT, etc.</p> <p>Set suppress where c1 eq missing or c2 eq missing or c3 eq missing endset rows all</p> <p>MISSING — Suppresses rows that contain only missing values.</p> <p>ZERO — Suppresses rows that contain only zeros.</p> <p>BOTH — Suppresses rows that contain zeros and missing values.</p> <p>FORMFEED — Suppresses the form feed at the top of a report, so that several reports can appear on one page.</p> <p>COLHEADING — Suppresses column headings defined in a COLHEADING-ENDCOLHEADING block. If you do not define a column heading block, SUPPRESS COLHEADING suppresses the default column headings, which are the selected members of the Across list.</p> <p>PARHEADING — Suppresses paragraph (partition) headings defined in a PARHEADING-ENDPARHEADING block. If you do not define a paragraph heading block, SUPPRESS PARHEADING suppresses the default paragraph headings, which are the selected members of the Down list.</p>
TABS	Inserts a tab character between columns, so that applications can use the output.

TABS SINGLE	If you do not override the tab character in the DISPLAY command, two tab characters are inserted in front of the long name. TABS SINGLE suppresses one of these tab characters, so that the long name is preceded by a single tab character.
TRUNCATE	Truncates column headers to the column width.
UNDERLINE	Specifies the width of the underlining triggered by UDATA.
WIDTHS	Specifies the width of the numeric columns. Each column can be up to 50 characters wide.
<list>	One or more integers separated by commas. Application Server displays an integer and an asterisk (*) before the width to indicate that the width will repeat that number of times. For example, 2*6 means two columns that are six characters wide. Groups of widths that repeat, for example, 9, 8, 9, 8, ..., can be enclosed in parentheses and preceded by a repetition count, as in 5(9, 8). You can also use DEFAULT instead of a numeric width. Note: A text variable's value can contain up to 255 characters. If a text variable's value contains more characters than the width specified here, Application Server truncates the variable's values to a width of <n>.
ZERO	You can use the following options:
<string>	Up to seven characters to use for true zero values.
BLANK	Inserts a blank for true zero values.
DEFAULT	Inserts a 0 for true zero values.
OFF	Removes any previously set true zero values.

Example 1

...This example suppresses rows with zero values, prints the column headers once, and indents the ... rows by two spaces. Column widths are set to 10, 10, 10, 10, 5, 8, and 8.

SET

```
SUPPRESS ZERO
COLUMNS ONCE
INDENT 2
WIDTHS 4*10, 5, 2*8
```

ENDSET

...This example sorts the last down dimension, and nested levels 1 and 2, based on column 1 values:

SET

```
STYLE LEVELTOTAL
SORT 1
SORT NLEVEL 1 1
SORT NLEVEL 2 1
```

ENDSET

...This example displays the top ten products sold in each region in the latest sales period:

SET

```
ACROSS Time
DOWN Variables, Region, Product
SORT LAST
```

ENDSET

SET-ENDSET (Report)

SCALAR i

DO i = 1, 10

 ROW[i]

ENDDO

Example 2

...This example, taken from a report called BASIC, prefaces each row with a number identifying

... whether the member is an input(1), output(2), or result(3). The numbers are useful for interpreting

...drill-down levels.

REPORT basic

SET

 TABS

 PREFACE '132'

ENDSET

ROWS ALL

DISPLAY basic

...The output of this command would be:

 TYPE Actual

 VARIABLES Sales

 Processor Chips

 Jan 00 Feb 00 Mar 00 Apr 00

2 West

2 Central

2 South

3 Total Region

1 New York

1 Boston

1 Chicago

1 Washington

1 Denver

1 Cleveland

1 St. Louis

1 San Francisco

1 Seattle

1 Los Angeles

1 Las Vegas

1 Dallas

Example 3

...This example, from a report called EXAMPLE, displays a report with default column headings:

SET

 COMMA ','

```

DECIMAL 0,1
ORDER 0, 1, 'mchg' = (c1-c2)%c2, 'ychg'=(c3-c4)%c4
TABS
NAME 20
ENDSET
ROWS ALL
DISPLAY EXAMPLE

```

...The output of this command would be:

```

286 Chip
      Jan 96  MCHG  YCHG
Cost of Sales      $3,288,567      $4.2      $3.1
Sales      $4,473,321      $1.7      $12.0

```

...From a report called EXAMPLE, display a report that suppresses the default column headings:

```

SET
  COMMA ','
DECIMAL 0,1
ORDER 0, 1, 'mchg' = (c1-c2)%c2, 'ychg'=(c3-c4)%c4
TABS
NAME 20
  SUPPRESS COLHEADING
ENDSET
ROWS ALL
  DISPLAY EXAMPLE

```

...The output of this command would be:

```

Cost of Sales      $3,288,567      $4.2      $3.1
Sales      $4,473,321      $1.7      $12.0

```

6.176 SET ENVIRONMENT

Use

Use the SET ENVIRONMENT command to set environment variables, such as DBHOME, DBPATH, or database path names.

Syntax

SET ENVIRONMENT <environment-variable-name> <value>

Parameter	Description
<environment-variable-name>	The name of the environment variable to set.
<value>	The value to set it to, including: DBHOME — The DBHOME variables define a directory in which Application Server creates new files by default. DBPATH — The DBPATH variables defines a colon-separated list of directories that Application Server searches for existing files. Application Server searches the current directory and the DBHOME directory (if specified) first.

SET FISCAL

Example: SET ENVIRONMENT

...Set the DBHOME and DBPATH environment variables in a UNIX Application Server session:

```
SET ENVIRONMENT DBHOME /lsserver/dbs
```

```
SET ENVIRONMENT DBPATH /lsserver/files/lsserver/install/dbs
```

6.177 SET FISCAL

Use

Defines the type of fiscal year for a database. The fiscal year is defined as one of three basic types:

CALENDAR — Specifies a fiscal year with 12 months that coincide with calendar months. This calendar is most appropriate when your database contains monthly variables. Do not use it if you intend to load daily or weekly data because you cannot specify a week start day. With the CALENDAR fiscal calendar, Application Server defines the starting day of the week according to the current SET PERIOD range. Consequently, the week start day will always be changing.

USER — Specifies a 12 month or 13 period (often called lunar) fiscal year. One or more additional keywords and options allow you to define when the year and months begin, and how weeks roll to months and months roll to quarters.

USER STARTING ENDING — Specifies a fiscal year of 12 periods that is completely defined by the user. (Referred to as a custom calendar in this documentation.)

Note: If you want to use biweekly measures, you must create a custom calendar. See the USER STARTING ENDING keyword below for details.

Syntax

SET FISCAL

```
{ CALENDAR [<month>] }
{ USER [<pattern>] [ANCHOR | CLOSE <dayofweek>] [<monthbegin>] [QUARTER <quarterpattern> ]
[<startmonth>] [<startday>] }
{ [<startmonth>] [<startday>] USER STARTING <datebegin> ENDING <dateend>, ..., <dateend> }
```

CALENDAR (Standard calendar)

Parameter	Description
<month>	The month the fiscal year starts in, for example, February. If you do not specify a month, the fiscal year begins in January.

USER

Parameter	Description
<pattern>	<p>Specifies how weeks roll into months or periods. Valid choices are 445, 454, 544, and 13. If no <pattern> is given, the option specified for <monthbegin> will determine how weeks roll into months.</p> <p>445 - Indicates that month one has four weeks, month two has four weeks, and month three has five weeks.</p> <p>454 - Indicates that month one has four weeks, month two has five weeks, and month three has four weeks.</p> <p>544 - Indicates that month one has five weeks, month two has four weeks, and month three has four weeks.</p> <p>These patterns repeat for each quarter.</p> <p>13 - Indicates that there are thirteen four-week or lunar periods per year. The way in which periods roll into quarters is set with the QUARTER <qtrpattern> keyword explained below.</p>

Note: A 445, 454, or 544 month pattern always begins in January. You need to allow for this if you specify a fiscal year start month that is not on a calendar year quarter break. For example, if your fiscal year begins in March and you want March to begin a 544 pattern, you have to set a 445 pattern instead as follows:

SET FISCAL USER 445 FIRST March

Pattern		Effect
4	Jan	
4	Feb	
5	Mar	
4	Apr	
4	May	
5	Jun	

ANCHOR	Specifies that the first day of the fiscal year will be the first day of the starting month and the last day of the fiscal year will be the last day of the ending month. If ANCHOR is omitted, <monthbegin> will determine the first and last days of the fiscal year.
CLOSE <dayofweek>	Specifies a day of the week before which the year must close. The actual close day is determined by the <startday> value. CLOSE can only be used with the <pattern> (445,13, and so on) options; it is not valid with the <monthbegin> (FIRST, LAST, MOST) option. This option is useful when your company calendar must end a number of days prior to the day your parent company ends its fiscal year. For example, SET FISCAL USER 445 CLOSE Thursday defines a fiscal year that starts in January and ends on the Saturday that precedes the last Thursday in December.
<monthbegin>	When used in conjunction with <monthpattern>, determines the start day of the first period of the year. When used in conjunction with ANCHOR, determines the start day of months 2 through 12 or 13. Valid choices are:
FIRST	The first day of the first week must fall in the calendar month. FIRST is the default if <monthbegin> is not specified.
LAST	The last day of the first week must fall in the calendar month. The year often begins in the previous calendar year.
MOST	The first week of the month that has most of its days fall in the calendar month determines the first day of the month. If four or more days in a week fall in the calendar month, then the week is included in that month; if less than four days fall in the calendar month, then the week is part of the previous month. Note: If you specify a <monthpattern> for a USER calendar then you can use either ANCHOR or <monthbegin> but not both. If you do not specify a <monthpattern> then ANCHOR and <monthbegin> can be used together. To state this is a different way, ANCHOR is used to set the begin and end of the fiscal year and <monthpattern> or <monthbegin> are used to set the start and end of all other months.
QUARTER <qtrpattern>	This option is valid only with the <monthpattern> option of 13. Specifies how lunar periods roll into quarters. Valid choices are 3343, 3334, 3433, and 4333. If no <qtrpattern> is specified, the default pattern is 3343.
<startmonth>	Specifies the calendar month that starts the fiscal year. The default is January.
<startday>	Specifies the start day for weeks. The default is Sunday.

USER STARTING ENDING (Custom calendar)

Parameter	Description
<startmonth>	Specifies the calendar month that starts the fiscal year. If you omit <startmonth>, January is used as the starting month. Application Server will not infer the start month from the start date that you specify in <datebegin>. At least half of the days of your first custom period must fall in the calendar month specified by <startmonth>. For example, if the first period in your custom calendar

SET FISCAL

	starts on August 20, 2006 and ends on September 16, 2006, September is the first month in your fiscal year.
<startday>	Specifies the day of the week that starts the fiscal year. If you omit <startday>, Sunday is used as the starting day. Application Server will not infer the start day from the start date that you specify in <datebegin>.
<datebegin>	Specifies the first day of the first year.
<dateend>, ..., <dateend>	A minimum of 72 month ending dates separated by commas. When you create a procedure to define a custom fiscal year, you set the first day of the first year and then a minimum of 72 month ending dates. More specifically, the procedure must define three years before and three years after the range of data stored in your database. If your database contains data for 2004 through 2007, then your customer calendar must start in the first month of 2001 and continue until the end of 2010. As you add additional months of data to your database, you can add more ending dates to your calendar procedure and re-execute it. This is the only time you can issue a SET FISCAL command when there are variables in your database. If you are using biweekly measures, you must define a custom calendar with dates for three years before and three years after the range of data stored in your database (72 months in addition to the dates for which there is data). Month definitions must consist of four- or six-week spans because a biweekly period cannot start in one month and end in another. If you define three biweekly periods in a month, you may get a warning about a "suspicious gap" between month start and month end dates. This is just a warning message and is acceptable. Note: If you do not properly define a custom fiscal year, Application Server displays the error "Attempt To Foresee Too Far Into The Future; More End-Of-Month Dates Required", whether the problem is missing dates at the beginning or the end of the period range.

Remarks

To maintain compatibility with earlier versions, three additional calendar types, described below, exist in Application Server. Each one has a corresponding setting in the calendars defined above.

CONSUMER	A synonym; the same as specifying USER 445 LAST.
RETAIL	A synonym; the same as specifying USER LAST.
THIRTEEN	A synonym; the same as specifying USER 13 LAST.

Example: SET FISCAL

...This example sets a calendar fiscal year beginning in April:

SET FISCAL Calendar April

...The following example sets a fiscal year beginning in January, with 13 lunar periods of 4 weeks in each period. The first quarter has four periods (sixteen weeks), the second quarter has three periods (twelve weeks), the third quarter has three periods (twelve weeks), and the fourth quarter has three periods (twelve weeks).

SET FISCAL USER 13 QUARTERS 4333 FIRST

...This example is similar, except that the fiscal year (and the quarterly pattern) begin in December instead of in January. The 4,3,3,3 pattern begins in December

SET FISCAL USER 13 QUARTERS 4333 FIRST DECEMBER

...This example defines a user fiscal year that coincides with a calendar fiscal year:

SET FISCAL User Starting Jan 1 06 Ending Jan 31 06, Feb 28 06, ..., Dec 31 06

...This example creates a fiscal calendar where fiscal August has five weeks because January starts the 454 pattern:

SET FISCAL USER 454 ANCHOR AUGUST SUNDAY

...This example creates a 13 period fiscal year where the periods start on Wednesday and the last period of the year must end before the last Saturday of the year (the close date):

SET FISCAL USER 13 CLOSE SATURDAY WEDNESDAY

...The procedure below defines a custom calendar that begins on the first Monday in February. Weeks roll into months and months roll into quarters following a non-standard pattern of 454-454-454-445

BEGIN

SET FISCAL February USER STARTING 02/05/96 ENDING

... fiscal year 96

03/03/96, 04/07/96, 05/05/96, 06/02/96, 07/07/96, 08/04/96,

09/01/96, 10/06/96, 11/03/96, 12/01/96, 12/29/96, 02/02/97,

... fiscal year 97

03/02/97, 04/06/97, 05/04/97, 06/01/97, 07/06/97, 08/03/97,

08/31/97, 10/05/97, 11/02/97, 11/30/97, 12/28/97, 02/01/98,

... fiscal year 98

03/01/98, 04/05/98, 05/03/98, 05/31/98, 07/05/98, 08/02/98,

08/30/98, 10/04/98, 11/01/98, 11/29/98, 12/27/98, 01/31/99

END

6.178 SET IGNOREMISSING MEMBERS

Use

Specifies the behavior of Application Server when you run a SELECT or ORDER command against a dimension that has missing members.

Syntax

SET IGNOREMISSING MEMBERS [ON | OFF]

Parameter	Description
ON	Default value. Instructs Server to issue a warning message listing the missing dimensional members, and to complete the SELECT or ORDER command.
OFF	Instructs Server to report missing dimensional members as an error and fail the SELECT or ORDER command.

6.179 SET INDENTATION

Use

SET INDENTATION specifies the number of blanks to indent a dimension's levels in output produced by a LIST command or a report. The indented dimension must be the last down dimension and must have been ordered using the ORDER HIERARCHY command.

Syntax

SET INDENTATION <n>

Parameter	Description
<n>	Specifies the number of blanks to indent, where <n> is an integer between 1 and 10. When you issue an ORDER <last_down_dimension> HIERARCHY command and a SET INDENTATION command and then you issue a LIST command or display a report, each level of the dimension is indented by the specified number of characters.

SET LABEL

6.180 SET LABEL

Use

SET LABEL defines the label set to be used for the specified dimension or variable.

Syntax

SET LABEL <name> [DEFAULT | SYNONYM <synonym>]

Parameter	Description
<name>	The name of the dimension or variable to which the labels are to be applied.
DEFAULT	Switches back to the standard Application Server long and short names for the specified dimension or variable.
SYNONYM <synonym>	The name of an Application Server synonym set containing alternative long and short names for the specified dimension or variable. Note: The synonym set cannot contain labels that begin with an asterisk and then a blank space. Labels can begin with other characters though. For example, you cannot use a label * units or * East but you can specify the label ~ units or ~ East.

Note: The syntax for the Hybrid OLAP SET LABEL (Schema) command is different.

6.181 SET LATEST|EARLIEST

Use

SET LATEST|EARLIEST defines a specified date as the date for latest or earliest available data.

Note: You cannot use EARLIEST in time sets.

Syntax

SET {LATEST | EARLIEST} {[<periodicity>] <date> | FROM <variable> [IN <dimension> <member> [..., IN <dimension> <member>]] } DEFAULT

Parameter	Description
LATEST	Sets the specified date as the latest date for available data.
EARLIEST	Sets the specified date as the earliest date for available data.
<periodicity>	Sets the data at one of the following periodicities: yearly, semiannually, quarterly, monthly, lunar, bimonthly, weekly, biweekly, daily, or hourly.
<date>	A date to use as the earliest or latest date for available data.
FROM <variable>	Sets the earliest or latest date based on the variable's earliest or latest date.
IN <dimension> <member>	Sets the date based on the variable's earliest or latest date within the specified dimension or member. Dimension member names that use special characters should be in single quotation marks (' ').
DEFAULT	Removes the latest/earliest setting from the database.

Example

...This example sets the earliest available monthly data at January

...1999 and the latest available monthly data at June 1999:

```
SET EARLIEST month 99/01
```

```
SET LATEST month 99/06
```

...This example sets the latest date based on the latest date found when scanning Sales data for the

... East region and actual data from the Type dimension:

```
SET LATEST FROM Sales IN Region East IN Type Actual
```

6.182 SET MEMORY

Use

SET MEMORY sets the number of bytes of memory for Application Server to use; however, you can specify the suffix M to denote megabytes (for example, 10M). By default, Application Server uses all available memory. The optimal amount of memory depends on your system

For commands such as CALCULATE and CONSOLIDATE, which process much or all of the database, you should set memory to a value less than your real memory size. Doing so may prevent unnecessary virtual storage (VS) paging.

Syntax

SET MEMORY <integer>

Parameter	Description
<integer>	Number between 0 and 999,999,999.

Note: You can control the default amount of available memory with the DEFAULTMEMORY environment variable. This specifies (in kilobytes only) the amount of memory you want to use per session by default. For best performance, both MEMORY and DEFAULTMEMORY should be set to somewhat less than the real memory available on your computer.

Using SET MEMORY

SET MEMORY default is the maximum virtual memory the given operating system allows the process, unless you set the environment variable DEFAULTMEMORY, in which case Application Server uses that value.

The setting for DEFAULTMEMORY is in kilobytes only — you cannot specify a suffix to denote megabytes. For example, DEFAULTMEMORY=4000 sets the environment variable to 4000KB.

For single-user local systems, or long-running overnight batch processes, such as a data load or consolidate, you should use the SET MEMORY command to set the maximum real memory less 'something' for the OS and the Application Server binaries and data. Trial and error might be useful for the batch processes. For Windows, 4-6MB should be enough to prevent excessive VS paging or swapping.

For multiple client/server sessions, the default setting should give fair performance, because a typical client is not likely to need a lot of memory for a long period, and because VS paging gives the best sharing of real memory between processes.

For batch jobs (load, consolidate, and so on), either use SET MEMORY or set the DEFAULTMEMORY environment variable, because some operating systems can terminate a process without issuing a message when the operating system runs out of swap space. This termination can occur under Windows and Window NT, AIX (RS600 UNIX), and other systems.

Example: SET MEMORY

...This example sets the memory to 40000 KB

SET MEMORY 40000

...You can use the suffix M to denote MB. This example sets the memory to 20 MB

SET MEMORY 20M

...which is the same as specifying 20000000 with no suffix

SET MEMORY 20000000

6.183 SET MESSAGE

Use

The SET MESSAGE command turns off the display of categorized messages. Some, but not all, messages have categorized message numbers associated with them (for example, DAT094). Application Server displays these numbers next to the message text in the dialog box. Application Server only displays these numbers for error messages. Application Server does not display these numbers for parsing errors.

Syntax

SET MESSAGE NUMBER|NONUMBER

Parameter	Description
NUMBER	Specifies that numbers should be displayed.
NONUMBER	Switches off the display of the message numbers.

Example: SET MESSAGE

...In this example, message numbers are set to be displayed, if they exist for a message. Two examples ... are given here, one with a number and one without.

SET MESSAGE NUMBER

ATTACH FRED

ENV040:

Database FRED is not registered as a valid database. Contact the Database Administrator for more information.

asghfdagshd

Expected an Application Server Command

...In this example, message numbers are turned off:

SET MESSAGE NO

ATTACH FRED

Database FRED is not registered as a valid database. Contact the Database Administrator for more information.

6.184 SET PERIOD

Use

SET PERIOD defines a date range that applies for all subsequent operations.

Syntax

```
SET PERIOD {<daterange> [HOUR <hourrange> ] }
           {<period> }
           {DEFAULT}
```

Parameter	Description
<daterange>	Range of dates in the form <yy[/mm[/dd]]> - <yy[/mm[/dd]]>. For example, 08/12/10 - 08/12/31 is the period from December 10, 2008 through December 31, 2008, and 08-00 is all of 2008 and 2009.
HOUR <hourrange>	Defines a range of hours in the form <hh:qq> - <hh:qq>, where: <hh> — Hour in a 24-hour clock: 0 through 23.

	<qq> — Quarter-hour: 00, 15, 30 or 45.
<period>	One of the following periods: yty — The latest period back to the equivalent period one year ago. yrago — The latest period one year ago. current — The period set with a SET LATEST command. previous — The previous period. latest — The period set with a SET LATEST command.
DEFAULT	Removes any period set with a previous SET PERIOD. Application Server uses either the period set for your variables or the period set in a statement with the PERIOD keyword. The PERIOD keyword overrides all other settings.

Example: SET PERIOD

...This example sets the period to June 10 through August 10, 1999:

SET PERIOD 99/6/10-99/8/10

...This example sets the period January 1999 to the period December 2000:

SET PERIOD 99-00

...This example sets the default period to the period after the end of the data:

SET PERIOD LATEST PLUS 1

...This example sets the period to January 99 through January 00:

SET LATEST Jan 00

SET PERIOD yty

...This example sets the period to January 00:

SET LATEST Jan 99

SET PERIOD yrago

...This example sets the period to December 00:

SET LATEST Jan 01

SET PERIOD PREVIOUS

...This example sets the period to January 00:

SET LATEST Jan 00

SET PERIOD LATEST

6.185 SET ONERROR

Use

SET ONERROR controls whether you want a procedure to continue or to stop running if a non-fatal error occurs during the procedure. Use this command before running the procedure.

Syntax

SET ONERROR { CONTINUE | CLEAR }

Parameter	Description
CONTINUE	Continues running the procedure until it is complete, even if errors occur.
CLEAR	Stops the procedure if any errors occur when running the procedure.

6.186 SET periodicity

Use

SET periodicity defines the periodicity to use for all subsequent operations; this periodicity overrides any default periodicities.

The default periodicity for a selection of variables is the periodicity with the largest scope. For example, if you select the variables Sales, Units, and Price, and the periodicity for Sales and Units is weekly, but the periodicity for Price is monthly, the default periodicity is monthly.

The specified periodicity applies to subsequent operations; it applies anywhere you want to override the default periodicity of the data.

Syntax

SET { <periodicity> | CONSTANT }

Parameter	Description
<periodicity>	Applies one of the following periodicities: yearly, ytd, ryearly, semiannually, quarterly, qtd, rquarterly, monthly, mtd, mytd, avgytd, rmonthly, lunar, bimonthly, weekly, wtd, biweekly, daily, hourly.
CONSTANT	Allows you to load data into a database that contains both constant and periodic measures. Note: Use the SET CONSTANT command prior to loading data into a variable defined as constant if the model contains both periodic and constant measures. It is only necessary if you first load into periodic measures and a periodicity has been explicitly or implicitly set for the first load.

6.187 SET PRINTER

Use

SET PRINTER defines the characteristics of a system printer. Normally, you use this command in a PROFILE procedure.

Syntax

SET PRINTER [WIDTH <width>] [LENGTH <length>] [UPPER]

Parameter	Description
WIDTH <width>	Defines the maximum number of characters that can print on a line, where <width> is the maximum number of characters.
LENGTH <length>	Defines the number of lines that can print on a page, where <length> is the maximum number of lines.
UPPER	Sets the printer to print only uppercase, alphabetic characters. Application Server translates all lowercase characters into uppercase.

Example: SET PRINTER

...This example specifies some printer characteristics:

SET PRINTER WIDTH 132 LENGTH 66 UPPER

6.188 SET PROGRESS

Use

SET PROGRESS turns off and on the progress meter displays from long-running commands, such as CALCULATE and CONSOLIDATE.

Syntax

SET PROGRESS {ON|OFF}

Parameter	Description
ON	Turns on the progress meter displays.
OFF	Suppresses the progress meter displays.

6.189 SET REASSURANCE

Use

SET REASSURANCE turns reassurance messages on and off. Reassurance messages are ongoing statements that inform you of what is occurring.

Syntax

SET REASSURANCE {ON | OFF}

Parameter	Description
ON	Turns on reassurance messages (default).
OFF	Turns off reassurance messages. When running an application, you should turn off the reassurance messages to avoid interference with the format of the output.

Example

...This example shows the effect of entering SET REASSURANCE OFF. Application Server no longer ... provides a Variables Selected message.

SELECT BY Merchandise

...The output of this command would be:

9 Variables Selected

SET REASSURANCE OFF

SELECT BY Merchandise, Version

...(no message appears)

6.190 SET REREAD

Use

SET REREAD rereads written blocks to the database, and validates that the contents arrived safely. Use this command only when you suspect I/O problems. A failed write causes the system to terminate abruptly.

Syntax

SET REREAD ON|OFF

Parameter	Description
ON OFF	Turns the reread function on and off.

6.191 SET QUERY VARIABLE

Use

This command sets a value for a Query Variable. It is similar to SET CONTROL.

SET SELECT

Syntax

SET QUERY [VARIABLE] [<LinkId>] [<Cube>] <Query Variable> <Query Value>

Parameter	Description
LINKID <LinkId>	Enter a specific LinkId if the Application Server database contains imported information from more than one Query and you want Query Variable information for just one Query Cube. If the Application Server database contains only imported information from a single Query, you do not need to supply a specific LinkId.
CUBE <cube>	Enter a specific cube if the Application Server database contains imported information from more than one Query and you want Query Variable information for just one Query Cube. If the Application Server database contains only imported information from a single Query, you do not need to supply a specific cube.
<Query Variable>	If a Query Variable is mandatory and has no default value, an application must supply a value for it before it asks the Connector for a view of data. Since Query Variable names must be in the ODBO format with enclosing [] and Query Variable names often begin with a numeric character, the Query Variable name should be enclosed in Quotes.

6.192 SET SELECT

Use

Specifies how the SELECT command resolves member names by expecting either short names (row names) or long names (labels).

Syntax

SET SELECT {NAME | LABEL}

Parameter	Description
NAME	Specifies that the SELECT command should look for short names only.
LABEL	Specifies that the SELECT command should look for both short and long names. This is the default value.

6.193 SET SHARE INTERVAL

Use

SET SHARE INTERVAL determines how often the system rechecks for file contention on shared files and database set contention on shared databases.

Syntax

SET SHARE INTERVAL <n>

Parameter	Description
<n>	Number of seconds between contention checks.

6.194 SET SHORT|LONG

Use

SET SHORT|LONG defines whether Application Server displays the variable and dimension member names or labels in lists and reports.

Syntax

SET {SHORT | LONG}

Parameter	Description
SHORT	Indicates that Application Server displays variable and dimension member names in lists and reports.
LONG	Indicates that Application Server displays variable and dimension member labels in lists and reports (default). If you do not define labels for variables or members, Application Server does not display their names.

Note: If you use the Access subsystem to import or export data, you must use variable and member names. If you have previously entered SET LONG, you must enter SET SHORT.

Use in the Dimensional Selector

In the Dimensional Selector, there are two modes of referencing the dimension member names — long names or short names. If you are using short names, you cannot enter a long name, and if you are using long names, you cannot enter a short name. The reason is that the short name for one member could be the same as the long name of another member. The Dimensional Selector cannot discern which you want.

Example

...Application Server displays variable names, rather than labels, in lists and reports:

SET SHORT

DISPLAY

6.195 SET SPARSE

Use

SET SPARSE changes variables stored as sparse so that they are stored as dense in a dumped database.

If this command is used before issuing a CREATE variable SPARSE command, the SPARSE keyword in the CREATE variable command will be ignored and the ON or OFF setting of the SET SPARSE command will be used.

If this command is used after a database has been dumped, any variable stored with the SPARSE keyword in the dump file will honor the setting of the SET SPARSE command during the LOAD.

Note: SPARSE is the better method for creating measures when databases are removed, recreated and populated from scratch each time and no additional data for extra periods is added to the series in that database. DENSE may be a better method for creating measures when additional data for extra periods is added because DENSE measures honor the MULTIPLES keyword, while SPARSE ones do not.

Syntax

SET SPARSE {ON | OFF}

Parameter	Description
ON	Confirms the CREATE variable SPARSE command and creates the variable as sparse.
OFF	Ignores the SPARSE keyword on any proceeding CREATE variable SPARSE commands.

Example

...This example would suppress the creation of sparse variables while loading DUMPFIL:

SET VARIABLES

SET SPARSE OFF

LOAD DUMPFIL

SET SPARSE ON

6.196 SET VARIABLES

Use

Assigns labels to variables, removes labels from variables, defines how Application Server converts a variable, and changes the default format for a variable.

Syntax

```
SET VARIABLES <variables>
  { FROM <linkid> PREFIX <prefixname> }
  { NOFROM }
  { LABELS '<labels>' }
  { DESCRIPTION <description> }
  { UNITS | NOUNITS }
  { RATE | NORATE }
  { EXPENSE | NOEXPENSE }
  { [INCLUSIVE | EXCLUSIVE [ZEROS]] [FIRST | LAST | AVERAGE | SUM | WEIGHTED
    <weighted_variable> | MIN | MAX] }
  { CONSOLIDATE | NOCONSOLIDATE }
  { CONVERT {BEFORE | AFTER } }
  {
    [ AFTER <c> | OFF ]
    [ AUTO [OFF] ]
    [ BEFORE <c> | OFF ]
    [ COMMA <c> | , | . | OFF | LOCALE ]
    [ CURRENCY <c> | $ | OFF ]
    [ DECIMALS <n> | LOCALE ]
    [ FIX CURRENCY | NEGATIVE | DECIMAL ]
    [ FLOAT CURRENCY | NEGATIVE | DECIMAL ]
    [ MISSING <string> | BLANK | DEFAULT ]
    [ NEGATIVE <prefix> [ <suffix> ]] | DEFAULT ]
    [ PAD [OFF] ]
    [ POINT <c> | . | , | LOCALE ]
    [ REVERSE [OFF] ]
    [ SCALE <n> | OFF ]
    [ WIDTH <n> ]
    [ ZERO <string> | - BLANK | DEFAULT | OFF ] }
```

Parameter	Description
<variables>	Names of one or more variables separated by commas. You can use an asterisk (*) instead of a name to indicate all variables. Variable names that use special characters should be in single quotation marks (' ').
FROM <linkid>	(Applies to Hybrid OLAP models.) Specifies a link identifier (<linkid>). Makes the selected variable a drillthru variable. Use the SET VAR variable_name NOFROM syntax to return the variable to a normal Application Server variable.
PREFIX <prefixname>	(Applies to Hybrid OLAP models.) Specifies a prefix string.
NOFROM	(Applies to Hybrid OLAP models.) Returns the selected drillthru variable to a normal Application Server variable
LABEL <labels>	Defines the labels you want associated with the variables, where <labels> is one or more labels of up to 255 characters, enclosed in single quotation marks (' ') and separated by commas. Labels correspond to variables based on their order in the list; that is, the first label is applied to the first variable, the second label is applied to the second variable, and so on.

Notes:

If you create labels for variables, the first 24 characters of each label must be unique.

Empty single quotation marks (' ') remove any previously set label.

You cannot create a label that begins with an asterisk and then a blank space. You can use other characters though. For example, you cannot specify '* units' but you can specify the label '# units'.

DESCRIPTION <description>	Defines the descriptions to associate with the variables, where <description> is the text that describes the variable. Empty single quotation marks (' ') remove any previously set description. Application Server displays the descriptions in the output from a SHOW VARIABLE command.
UNITS NOUNITS	Specify UNITS to indicate that Application Server should round a variable's values to the nearest integer when it uses the variable at a periodicity other than its original one. You can turn off the UNITS option with the NOUNITS keyword. NOUNITS is the default.
RATE NORATE	Specify RATE to define the variables as rate variables. Application Server averages the variables over time, but does not consolidate them at the output level of dimensions. A rate variable is often used with percent or price variables. You can turn off the RATE option with the NORATE keyword. NORATE is the default setting.
EXPENSE NOEXPENSE	Specify EXPENSE to define the variables as expenses. You can develop a conditional logic set to test on this attribute and then treat the variable differently for variance calculation. You can turn off the EXPENSE option with the NOEXPENSE keyword. NOEXPENSE is the default setting.
INCLUSIVE EXCLUSIVE [ZEROS]	INCLUSIVE means that missing values are included in the conversion. EXCLUSIVE means that missing values are excluded from the conversion. ZEROS treats zeros as missing values. Use this to treat data from BW properly by excluding zero values from the conversion.
AVERAGE	Uses the average value of the variable in the time series when Application Server consolidates the variable over time (the default for rate variables). An INCLUSIVE AVERAGE variable will count a missing time period as zero. An EXCLUSIVE AVERAGE variable will ignore missing values when calculating the average value for the variable.
SUM	Uses the sum of the values in the time series when Application Server consolidates the variable over time (the default for all variables, except rate and text).
WEIGHTED <weighted_variable>	Uses the weighted average of the variable. For example, price might be weighted by units sold. The quarterly price for monthly data is the sum of price multiplied by the number of units sold during the months, divided by the sum of the units sold for those same three months.
MIN	Uses the minimum value of the variable in the time series when Application Server consolidates the variable over time. For example if you view a monthly measure quarterly and the measure is MIN, you see the smallest monthly value as the quarterly value.
MAX	Uses the maximum value of the variable in the time series when Application Server consolidates the variable over time.
CONSOLIDATE NOCONSOLIDATE	Specifies CONSOLIDATE to indicate that Application Server should consolidate the variable. Specify NOCONSOLIDATE to indicate that Application Server should not consolidate variables when you enter a CONSOLIDATE or CALCULATE command. This function is useful when you have already consolidated the data, or when you are calculating variables based on already consolidated data. CONSOLIDATE is the default setting.
CONVERT BEFORE AFTER	Specifies whether the calculation of output members in the time set occurs BEFORE or AFTER the calculation of virtual measures. AFTER is the default behavior for a virtual measure.

SET VARIABLES

Notes:

If you have virtual measures calculated from other virtual measures, you will need to make sure that all of them have the same SET VARIABLE CONVERT settings.

You can also change the order of calculations when a virtual variable is used in conjunction with a time set with output calculations. Because the SET VARIABLE command requires exclusive access to the dimensional model, use the VIRTUAL CONVERT when you do not have exclusive access.

If a VIRTUALCONVERT statement exists in the time set, it overrides any SET VARIABLE CONVERT settings.

AFTER '<c>'	A string of up to three characters to appear after values.
AFTER OFF	Removes any previously set string of characters.
AUTO	Performs automatic scaling.
AUTO OFF	Turns off automatic scaling.
BEFORE '<c>'	A string of up to three characters to appear before values.
BEFORE OFF	Removes any previously set string of characters.
COMMA '<c>'	A character to insert as the thousands separator.
COMMA ,	Inserts a comma as the thousands separator.
COMMA .	Inserts a period as the thousands separator.
COMMA OFF	Removes any previously set thousands separator.
COMMA LOCALE	Sets the thousands separator to the character defined in the regional settings in the Windows Control Panel.
CURRENCY '<c>'	A string of up to four characters to insert as the currency symbol.
CURRENCY \$	Inserts a dollar sign (\$) as the currency symbol.
CURRENCY OFF	Removes any previously set currency symbol.
DECIMALS <n>	Specifies the number of decimal places Application Server displays for the variable.
DECIMALS LOCALE	Specifies that Application Server should display values with the number of decimal places specified in the regional settings of the Windows Control Panel.
FIX CURRENCY	Places the currency symbol in the leftmost position.
FIX NEGATIVE	Places the negative sign in the leftmost position.
FIX DECIMAL	Places the decimal point symbol in the leftmost position.
FLOAT CURRENCY	Places the currency symbol adjacent to the value.
FLOAT NEGATIVE	Places the negative sign adjacent to the value.
FLOAT DECIMAL	Places the decimal point symbol adjacent to the value.
MISSING '<string>'	A string of characters to insert for missing values.
MISSING BLANK	Inserts a blank for missing values.
MISSING DEFAULT	Inserts a dash (-) for missing values.
NEGATIVE <prefix>	A character to appear before a negative number. If you specify a prefix without a suffix, enclose it in single quotation marks (' '). For example, NEGATIVE '('.
NEGATIVE <suffix>	A character to appear after a negative number. Enclose the suffix in single quotation marks (' '). For example, NEGATIVE ')' displays negative numbers as (2,123) instead of as -2,123.
NEGATIVE DEFAULT	Inserts a negative sign (-) before a negative number.
PAD	Pads numbers with leading zeros.
PAD OFF	Removes any padding.
POINT '<c>'	A character to use for the decimal point.
POINT .	Uses a period (.) as the decimal point.
POINT ,	Uses a comma (,) as the decimal point.

POINT LOCALE	Sets the decimal point to the character defined in the locale settings in the Windows Control Panel.
REVERSE	Reverses the position of currency and negative symbols. The default is the currency symbol followed by the negative symbol, for example, \$-2.00.
REVERSE OFF	Returns the currency and negative symbols to the previous positions.
SCALE <n>	Scaling factor; divides the number to be displayed by 10 to the power <n>. For example, scale 2 divides the number by ten to the second power, rounding down halves. That means that when you use scale 2, 2.505 is rounded to 2 decimal places to 2.50, not 2.51.
SCALE OFF	Removes any previously set scaling factor.
WIDTH <n>	Number specifying the field width. The maximum width that you can set for a text variable is 50. Note: A text variable's value can contain up to 255 characters. If a text variable's value contains more characters than the width specified in WIDTH <n>, Application Server truncates the variable's values to a width of <n>.
ZERO <string>	Up to seven characters to use for true zero values.
ZERO BLANK	Inserts a blank for true zero values.
ZERO DEFAULT	Inserts a dash (-) for true zero values.
ZERO OFF	Removes any previously set true zero values.

Remarks

The SET DEFAULT command specifies the default global formatting settings for all variables. However, if you use SET VARIABLES to change a formatting option for a variable, that variable will no longer use the default global formatting settings specified by SET DEFAULT.

To compare the default global formatting settings with the current formatting settings for a variable, use the SHOW DEFAULT and EXHIBIT VARIABLE varname commands.

If you use SET VARIABLES to change the formatting for a variable, you must use the REMOVEFORMATvariable command if you want to reset the variable so that it uses the global formatting settings.

Example: SET VARIABLES

...This example defines the variable Price as a rate:

```
SET VARIABLES Price RATE
```

...This example associates labels with the variables Sales and Profit. Application Server prints the ... labels instead of the variable names for lists and reports.

```
SET VARIABLES Sales, Profit LABELS 'Yearly Sales', 'Yearly Profits'
```

...This example removes the label from the variable Profit:

```
SET VARIABLES Profit LABELS ''
```

...This example assigns a description to the variable Sales:

```
BEGIN
```

```
SET VARIABLES Sales DESCRIPTION
```

Sales are collected daily from POS terminals and consolidated into weekly figures on the central computer for analysis.

```
END
```

...This example assigns a width of eight characters without any decimal places to all variables except ... Pct_Change, which is assigned a width of six characters and two decimal places:

SHOW

SET DEFAULT WIDTH 8 DECIMALS 0**SET VARIABLES Pct_Change WIDTH 6 DECIMALS 2**

The following two examples show the different outputs generated with the keywords CONVERT BEFORE and CONVERT AFTER, using a time set that calculates quarterly, and a virtual measure (vv1) that equals v1/v2.

SET VARIABLE <variable> CONVERT BEFORE gives this result:

	V1	V2	VV1
Jan 99	60.00	6.00	10.00
Feb 99	60.00	6.00	10.00
Mar 99	60.00	6.00	10.00
Q1	180.00	18.00	10.00

SET VARIABLE <variable> CONVERT AFTER gives this result:

	V1	V2	VV1
Jan 99	60.00	6.00	10.00
Feb 99	60.00	6.00	10.00
Mar 99	60.00	6.00	10.00
Q1	180.00	18.00	30.00

6.197 SHOW

Use

SHOW displays information about the current Application Server session. Specify SHOW before each keyword in the syntax above. For example, you can use SHOW ACROSS and SHOW DOWN but you cannot use SHOW ACROSS DOWN.

Syntax

```
SHOW
{ACROSS [ DETAIL | SPAN ] [TABS] [NOHEADER] [NONUMBER] [FORMAT <format> ] }
{DOWN [ DETAIL | SPAN ] [TABS] [NOHEADER] [NONUMBER] [FORMAT <format> ] }
{ATTRIBUTE}
{CALENDAR <year>}
{CONTROL [<pattern> ]}
{CORRECTIONS [FULL]}
{CUSTOM [ALL | FULL | TABS] [ [<database>].[<owner>.<dimension>.<setname>]
{DATABASE [<database> ]}
{DBSTATS}
{DEFAULT}
{DISORDERED <dimension>}
{DUPLICATES <dimension> [REPLACE]}
{ELAPSED}
{FORMAT}
{LABELS [<variablelist> ]}
{LATEST}
{EARLIEST}
{ORPHANS <dimension>}
{STEPCHILDREN <dimension>}
{SELECTED [<dimension> ]}
{SETTINGS}
{<settype>}
```



```

{<setname>}
{SPAN [DETAIL [NOHEADER] [NONUMBERS] [TABS] [FORMAT <format>]] }
{SPARSITY [<variablelist> | BY <dimensionlist> ] [EXCLUDE]}
{VARIABLES <variablelist>} [EXCLUDE]}
{UPDATED [NOHEADER] [TABS] [UTC [<variable>]] }
{VARIABLES [ BY <dimlist>]}
{VERSIONS}
{[EXACTLY] WHERE [<settype>] <name> [<member>][,.. <name> [<member>]]}

```

Common Parameter	Description
TABS	Delimits output with a tab. If you do not use TABS, the output is displayed with space characters between columns.
NOHEADER	Displays output with no heading line.
NONUMBER	Displays output with no line numbers.
FORMAT <format>	Specifies the format to display the dates. <Format> can be: myy, ymd, yynd, dmyy, dmy, mdyy, mdy, ydm, yydm, yym, my, or ym If the year is four digits (2010), the formats with yy are used. If the year is two digits (10), the formats with y are used. If the month is alphabetic (May 2010), the letter m is replaced with the word month, for example monthyy. The separator characters dash (-), slash(/), or period (.) can be used in the date; however, they must appear in the actual date format. For example, the format for 12-3-2010 would be m-d-y. You can use the string 'month' in place of 'm', in which case the month will be name of the month rather than the number. For example, the format for Dec/3/10 would be Month/d/y.
UTC <variable>	Displays any date in the UTC format of YYYYMMDDHHMMSS. If you specify a variable, it displays the date in UTC format for a specific variable. If you do not specify a variable, it displays the dates in UTC format for all variables. YYYY is the year, MM is the month, DD is the day, HH is the hour, MM is minutes and SS is seconds.

Parameter	Description
ACROSS	Displays current across settings.
DETAIL	Displays members that make up each column of the across setting. Use this with ORDER in reports. When you use SHOW ACROSS DETAIL, you should minimize the number of dimension members in the down list for performance reasons. To do this, select one member from each down dimension; this does not change your across list selections.
SPAN	Displays current across settings and also a time span for the TIME or a time set. For a time set, the time span will show the span of any INPUTS, and for OUTPUTS will show the span corresponding to the all encompassing span that results from the union of all the spans of the INPUT descendants of each OUTPUT. The spans will take the SET LATEST into account. For DAILY data, if any span is a single day, only the start of the span will be shown (i.e. no range just a single date). The analysis of the time set to determine the input descendants of OUTPUTs in time sets is done during COMPILE TIME.
DOWN	Displays the current down settings.
DETAIL	Displays members that make up each column of the down setting.
SPAN	Displays current across settings and also a time span for the TIME or a time set. For a time set, the time span will show the span of any INPUTS, and for OUTPUTS will show the span corresponding to the all encompassing span that results from the union of all the spans of the INPUT descendants of each OUTPUT. The spans will take the SET LATEST into account. For DAILY data, if any span is a single day, only the start of the span will be shown (i.e. no range just a single date).

SHOW

	The analysis of the time set to determine the input descendants of OUTPUTs in time sets is done during COMPILE TIME.										
ATTRIBUTE	Lists the attribute sets defined in the database.										
CALENDAR <year>	Displays the fiscal calendar for a specific year, for the current Use database, where <year> is the specific year.										
CONTROL	Displays control variables.										
<pattern>	Specifies a specific control variable name or wildcard pattern. If you omit <pattern>, this command displays all control variables. Four standard control variables always exist: LIBRARYDATABASE, SID, USEDATABASE, and WORKDATABASE, for example:										
	<table> <tr> <th>Control Variable</th><th>Value</th></tr> <tr> <td>LIBRARYDATABASE</td><td>TBDB</td></tr> <tr> <td>SID</td><td>Windows 95</td></tr> <tr> <td>USEDATABASE</td><td>JUICE</td></tr> <tr> <td>WORKDATABASE</td><td>WKADMIN</td></tr> </table>	Control Variable	Value	LIBRARYDATABASE	TBDB	SID	Windows 95	USEDATABASE	JUICE	WORKDATABASE	WKADMIN
Control Variable	Value										
LIBRARYDATABASE	TBDB										
SID	Windows 95										
USEDATABASE	JUICE										
WORKDATABASE	WKADMIN										
CORRECTIONS [FULL]	<p>Lists all members of a dimension that require automatic multiple counting corrections together with a list of each ancestor that requires adjustment and a correction factor.</p> <p>To compare with SAP BW data, if you look at the Hierarchy Structure Table for a hierarchy, you should see the same correction factors that you see in Application Server in the output from SHOW CORRECTIONS.</p> <p>FULL shows the correction factors and for each member that has correction factors, it displays hierarchically the ancestors of that member to illustrate where the multiple counting occurs.</p> <p>Note: When you compile a dimension or attribute, the system searches for multiple counting issues that arise if a dimension's members have multiple parents. During aggregation, the system corrects multiple counting issues by creating adjustments to eliminate them. The adjustments are used during a CONSOLIDATE.</p>										
CUSTOM	Shows details about currently defined User-Defined Hierarchies. The default information includes the database name, owner name, dimension name, setname, whether the User-Defined Hierarchy was saved with the PUBLIC or PRIVATE keyword, and the saved ID.										
ALL	Displays details about all saved User-Defined Hierarchies.										
FULL	Displays extra columns for the creator of the User-Defined Hierarchy, the last user to update the saved User-Defined Hierarchy, the time of creation, and the last update time.										
<database>	<p>Name of the database containing the User-Defined Hierarchy whose details you want to display. If you omit the database name, the current USE database is used.</p> <p>Note: You can use wildcards in any of the values for database, owner, dimension, and setname. For example, REMOVE CUSTOM J*.A*.Product.* If you omit any of the values, the wildcard * is implied.</p>										
<owner>	Name of the Application Server user who is responsible for the User-Defined Hierarchy. This name was specified during the SAVE CUSTOM command when the User-Defined Hierarchy was created. If you omit the owner name, the current Application Server user name is used.										
<dimension>	Name of the dimension whose User-Defined Hierarchy you want to review.										
<setname>	Name of the procedure set where the User-Defined Hierarchy was saved during the SAVE CUSTOM command.										
DATABASE	Provides summary information about all attached databases.										
<database>	Specifies the name of a specific attached database you want to get information about.										

DBSTATS	Displays statistical information about the current database and buffer setting recommendations.
DEFAULT	Displays default formatting values for column width, decimal places, currency symbol, and automatic scaling.
DISORDERED <dimension>	Displays output members in <dimension> that are not in the correct order for the Rollup subsystem to use. For example, you might have declared level 3 members before level 2 members.
DUPLICATES <dimension>	Displays members and labels that have the same value within the <dimension>.
REPLACE	Specifies that any duplicate member is replaced by <longname> (<shortname>). Only the duplicates are replaced. If a member is not a duplicate, it remains as is. After you issue this command with the REPLACE keyword, you see a message notifying you to compile the dimension. You must issue a COMPILE DIMENSION <dimension> command for any dimension whose duplicate members you replaced.
ELAPSED	When this command is first used, Application Server displays the number of seconds that have elapsed since login; with each additional use, Application Server displays the number of seconds that have elapsed since the last time you entered the SHOW ELAPSED command.
FORMAT	Displays current variable formats, as specified with SET DEFAULT or SET <variables>.
LABELS	Displays labels associated with all variables.
<variablelist>	Rather than display labels for all variables, you can specify a list of variables you want to display labels for. Separate each variable with a comma. You can use an asterisk (*) in a variable name to indicate that any character can occupy that position or any of the remaining positions in the name.
LATEST	Displays the data point you have set with SET LATEST.
EARLIEST	Displays the data point you have set with SET EARLIEST.
ORPHANS <dimension>	Shows the input and output members that do not have a parent (do not belong to a hierarchy and are not consolidated) within the specified dimension.
STEPCHILDREN <dimension>	Shows the input members that have more than one parent (belong to more than one hierarchy) within the specified dimension.
SELECTED [<dimension>]}	Displays currently selected variables, or the selected members of a dimension. Omit the dimension parameter to display selected variables, or specify the name of a dimension to display its selected members.
SETTINGS	Displays system information, such as SHARE INTERVAL, process ID, and client/server information.
<settype>	Lists the names of all the sets of that specified type in the database, where <settype> can be either TIME, LOGIC, REPORT, PROCEDURE, DOCUMENT, or DIMENSION.
<setname>	Displays information about the specified set. The parameter <setname> cannot be a procedure name or a document name, but it can be one of the following: <timeset> — Lists the inputs of the specified time set. <logic> — Lists the dimensions and the variables referenced by this logic set. <report> — Shows information about the requested rows and referenced external functions. <dimension> — Lists the input members, output member, and result member. Application Server displays excluded members with parentheses around them.
SPAN	Shows the start and end date that will be used in the current view. When a time set is involved it will show the dates based on the intersection of the current SET PERIOD and the periods implied by the time set, taking all the LATEST and EARLIEST settings into account. Note: Where the time set implies a period range outside the SET PERIOD range, SHOW SPAN produces no output. That is, SHOW SPAN only works on the current periodicity and period.

SHOW

DETAIL	Shows the start and end date of each time period that will be used in the current view. For example, if the data is monthly, the start and end date will be listed and each month between the start and end date. This view includes column headings of "Member", "Time", and "Dimension Members".
SPARSITY	<p>Shows the sparsity of time series on a variable-by-variable and quadrant-by-quadrant basis. Specifically, it shows the following information:</p> <p>Variable being analyzed</p> <p>Whether the variable has the SPARSE setting selected (either SPARSE=TRUE or SPARSE=NO)</p> <p>Quadrant-by-quadrant breakdown of the bytes used by SPARSE and non-SPARSE storage.</p> <p>Total number of series, total bytes used if the variable is set to SPARSE, total bytes used if the variable is set to not SPARSE, and a breakdown in 10% ranges of the counts of series where: $(\text{SPARSE_SIZE}/\text{NON_SPARSE_SIZE}) * 100$ fits in each range.</p> <p>This lets you determine the impact on the size of the database if a variable has been created as sparse but is actually dense, or if a variable has been created as dense but is actually sparse. A variable stored as sparse might occupy more space in the database than if it is set to dense.</p> <p>If you learn that a measure created as dense can occupy much less space if it were stored as sparse, then you can potentially significantly reduce the size of your database and improve the load and consolidate times of the database by creating that measure as a SPARSE measure.</p> <p>You may find that a SPARSE measure might use more space than a non-SPARSE measure only where all the series for that measure have different values in every time period. For example, the JUICE database has all the data for all the measures exists for all the time periods.</p> <p>Also note that SPARSE measures ignore the MULTIPLES setting for your database. This is another reason why SPARSE measures occupy less space than non-SPARSE ones, because they do not allocate any extra space in any time series for the addition of data in prior or later periods. If you intend to repeatedly add data for extra periods to a measure, it might still prove to be beneficial to have it declared non-SPARSE even when SHOW SPARSITY shows that will consume more space, because this will lead to less fragmentation in the database as extra data is added.</p>
<variablelist>	One or more variable names to analyze, separated by commas.
BY <dimensionlist>	One or more dimensions separated by commas.
EXCLUDE	Displays sparsity information about the specified variables, but omits information about hidden variables.
VARIABLES <variablelist>	<p>Displays information about one or specific variables in the Use database: the periodicity, the date range, and the dimensions with which they are associated. For the <variablelist> parameter, you can specify the name of one or more variable names, separated by commas.</p> <p>Note: If SHOW VARIABLES finds a distributed variable, an extra line is displayed before the variable details, showing the location of the variable.</p>
EXCLUDE	Displays information about the specified variables, but omits information about hidden variables.
UPDATE	Displays database variables and the date and time Application Server last updated them.
VARIABLES	<p>Displays information about the variables in the Use database: the periodicity, the date range, and the dimensions with which they are associated. SHOW VARIABLES also displays information about temporary variables created by the CREATE <variable> TEMPORARY command. Application Server stores temporary variables in the Work database.</p> <p>Note: If SHOW VARIABLES finds a distributed variable, an extra line is displayed before the variable details, showing the location of the variable.</p>

- BY <dimlist> Limits the number of variables shown by specifying only those with specific dimensions. To do this, enter SHOW VARIABLES BY <dimlist>, replacing <dimlist> with the names of dimensions separated by commas.
- VERSIONS Displays versions in the database.
- [EXACTLY] WHERE [<settype>] <name> [<member>] [... <name> [<member>]] Displays the locations of the variable, procedure, logic set, dimension, or other names in the database. For example, after removing a variable from Application Server, you can use this command to display all references to that variable. Remove these references to avoid a future reference to a nonexistent variable.

6.198 SHOW (Access External)

Use

SHOW is an Access External command that displays the current description statements, if specified. Otherwise, the default descriptions corresponding to ACROSS and DOWN appear.

Syntax

SHOW

6.199 SHOW (Supervisor)

Use

SHOW is a Supervisor command that displays user and database information.

Syntax

```
SHOW { DATABASES [<database> [..., <database>]] }
      { LOGINS [<user> [..., <user>]] }
      { GROUPS [<user_defined_hierarchy>] }
      { USERS [<user> [..., <user>]] }
```

Parameter	Description
DATABASES	Displays information about databases and their partitions.
<database>	Shows information about a specific database, including partitioned information, where <database> is one or more database names separated by commas.
LOGINS	Displays information about users currently logged in to Application Server, or those temporarily paused with EXIT.
<user>	Names of one or more users separated by commas. If you do not specify a name, Application Server displays information about all users.
GROUPS	Lists all User-Defined Hierarchies. Note: You define User-Defined Hierarchies to restrict user access to specific dimensions.
<user_defined_hierarchy>	Lists all users that belong to the specified User-Defined Hierarchies.
USERS	Displays information about all users in the database.
<user>	Displays information about a specific user, where <user> is one or more user names, separated by commas.

Example: SHOW

...This example shows the default variable settings:

SHOW DEFAULT

...The output of this command would be:

Width 12 Decimals 0 Comma ',' Auto

SHOW (Supervisor)

...This example shows the variable settings for the specified
...variable and the dimensions by which the variable is dimensioned:

SHOW FORMAT Sales

COUNTRY:

SALES

Width 10 Decimals 2 Currency "\$" Auto

SHOW DATABASE bank

USE Database: **BANK** Maximum Observations: 250

	Access	Maxblocks	% Available	Mode
BANK	UPDATE	200	49%	EXCLUSIVE

SHOW VARIABLES

PRODUCT:

VariableType	Items	Date Range
ADVERTISING	NU 12	MO Jan 1999 - Dec 1999
EXPENSES	NU 12	MO Jan 1999 - Dec 1999
PRICE	NU 6 BM	Jan 1997 - Dec 1999

COMPANY:

VariableType	Items	Date Range
REVENUE	NU 12	MO Jan 1997 - Dec 2000
SALES	NU 12 MO	Jan 1997 - Dec 2000

SHOW DIMENSION Product

PRODUCT

Inputs

ASPIRIN	ANTACIDS	ROLL ON	CREAM SPRAY
PERSONAL	LAUNDRY	DISHWASHING	

Outputs

SOAPS	DEODORANTS	HYGIENE	GENERIC DRUGS
--------------	-------------------	----------------	----------------------

Result

PRODUCT LINE

SHOW LOGIC Rest

REST

Database Variables

REVENUE	SALES VOL	UNIT PRICE	COST
UNIT COST	CONTRIB	NEWCON	

Referenced External Functions

MARGIN_CALC

SHOW SELECTED

VARIABLES

Inputs

UNIT PRICE	SALES	REVENUE	NET PROFIT
-------------------	--------------	----------------	-------------------

SHOW WHERE VARIABLE Sales

SALES Appears In

Logic BUDLOG, REVLOG

Report BUDREP, ANNREP, REVREP

SHOW EXACTLY WHERE 'USA'

USA Appears in

Logic REVLOG

20: case USA

50: 'USA ratio' = 3.50

SHOW WHERE S s

S# Appears In

Logic NEWLOG

SORTS Appears In

Logic SORTLOG

.

. *other names beginning with s*

.

...This example shows the output of the **DOWN DETAIL** option:

SHOW DOWN DETAIL

...The output of this command would be:

Row DOWN Dimension Members

1 Orient, COSTS

2 Orient, SALES

3 Orient, PROFIT

4 Orient, WORKTEXT

5 Middle, COSTS

6 Middle, SALES

7 Middle, PROFIT

8 Middle, WORKTEXT

9 Occident, COSTS

10 Occident, SALES

11 Occident, PROFIT

12 Occident, WORKTEXT

13 A_35_character_name_1234567890abcde, COSTS

14 A_35_character_name_1234567890abcde, SALES

15 A_35_character_name_1234567890abcde, PROFIT

16 A_35_character_name_1234567890abcde, WORKTEXT

...This example shows the standard, read-only control variables:

SHOW CONTROL

...The output of this command would be:

SHOW (Supervisor)

Control Variable Value

```

-----
LIBRARYDATABASE  TBDB
SID      Windows 95
USEDATABASE      JUICE
WORKDATABASE     WKADMIN

```

...This example shows how to tell if a user pressed an OK or Cancel button in the Dimensional Selector. The Application Server command **SHOW CONTROL** will display two variables that the Dimensional Selector creates (if the user presses OK), called **SELECT_ITEMS** and **SELECT_NUMBERS**. The user can check for those items before invoking the Dimensional Selector:

SHOW CONTROL

...clear them if they exist:

CLEAR SELECT_ITEMS**CLEAR SELECT_NUMBERS**

...and then check them again after the Dimensional Selector has finished. If the user cancelled, the control variables won't exist. If the user pressed the OK button, the control variables will exist.

...This example displays the calendar for the year 2000:

SHOW CALENDAR 2000**Example**

...This example lists all User-Defined Hierarchies in MASTERDB:

SHOW GROUPS

...The output of this command would be:

```

FINANCE
PRODMGR
SALESMGR

```

...This example lists all users in the Finance User-Defined Hierarchy:

SHOW GROUP Finance

...The output of this command would be:

```

ADMIN
DAVE
BETTE

```

...This example displays information for user Fin1:

SHOW USERS Fin1

...The output of this command would be:

```

User: FIN1Use Database: BUDGET Temp Database: FIN1TEMPBlocks: 300 Blksize 2048Buffers: 300
Library Database: TBDBSecurity Level: 99 Login State: OnLast Login: Tue Sep 12 11:21:48 2000

```

...This example displays information for all users who are currently logged in or paused:

SHOW LOGINS

...The output of this command would be:

ROBERTO: Paused FIN1: Logged In MARKETING: Paused

...This example displays information about the database, JUICE:

SHOW DATABASE JUICE

...The output of this command would be:

Database: JUICE (C:\PROGRA~1\PILOT~1\DATA\JUICE)

Maxaccess: UPDATE Disposition: ENABLED Default Usage: Exclusive

Current State: AVAILABLE Protection Key: None

Last user: ADMIN Last access: Fri Sep 15 14:30:07 2000

Blksize: 8192 Observations 1000 Maximum Members: 1000

Max. Size of the Database is 32 Gigabytes.

Maxblocks: 5000 Number Free: 4215

...The following example lists the partitions for the ACCNT database:

SHOW DATABASE Accnt

...The output of this command would be:

MASTERDB (C:\PROGRA~1\PILOT~1\DATA\MASTERDB) Maxblocks 100 Number Free 65Database:
 ACCNT Maxaccess: UPDATE Disposition: ENABLED Current State: AVAILABLE Protection
 Key: Non Last user: ADMIN Last access: Fri Sep 8 15:38:21 2000 Blksize: 8192 Observations
 500 Maxblocks: 120000 Number Free: 119988Database ACCNT Partition Information.Number of
 Partitions: 5Blksize: 8192Date Last Opened for Update: Fri Sep 8
 15:36:21 2000 Date Last Closed After Update: Fri Sep 8 15:36:21 2000 Partition Size Name
 1 20000 ACCNT01 2 20000 ACCNT02 3 20000 ACCNT03 4 40000
 ACCNT04 5 20000 ACCNT05

6.200 SKIP (Report)

Use

SKIP is a Report command that prints a blank line or lines.

Syntax

SKIP [<n>]

Parameter	Description
<n>	Number of blank lines you want printed. The default is 1.

Example: SKIP

...This example skips one line between the Sales and Revenue rows:

ROWS Sales

SKIP

ROWS Revenue

6.201 SNAPSHOT

Use

SNAPSHOT produces a subset of a database in a dump file, suitable for loading into another database on the same machine or on a different machine at a remote location.

SNAPSHOT

Syntax

SNAPSHOT <file> [SIZES <sizes>] [UPDATE] [IDENTICAL] [DUMMY] [STATUS <procedure>] [PRUNE]
 [PERIOD <range>] [EXPAND | REDUCE] [OVERWRITE] [DRILLTHRU] [NOATTRIBUTE] [NODENSE]
 [NOLARGEFILES]

Parameter	Description
<file>	Specifies the name of the file into which you want to place the dumped information. Note: By default, one dump file will be created. The file can exceed 2Gb. If you don't want the dump file to exceed 2Gb, you can partition multiple snapshot files by using wildcard characters to append the filename with numbers and using the NOLARGEFILES and/or SIZES keywords.
SIZES <sizes>	<p>Specifies the size of the file(s) that will contain the dumped information. To control the sizes of multiple partitions, specify a list of sizes separated by commas. If you specify fewer sizes than there are partitions, the last value is used as the size for the remaining partitions.</p> <p>Note: When specifying the SIZES parameter, the values cannot be greater than 2Gb if expressed as raw (unscaled numbers). To specify a SIZE greater than 2Gb you must use the K, M, or G scaling factors.</p> <p>You can specify the sizes in any of the following units:</p> <p>Bytes, for example 2000000000 Kilobytes, for example 2500000K Megabytes, for example 5000M Gigabytes, for example 3G</p>
NODENSE	If you want to snap a database and make sure it is backward compatible so that it can be loaded in a version of Application Server prior to version 9.4, you should use the NODENSE keyword. This prevents the snap file containing explicit commands to create DENSE measures, as this keyword would be rejected in any versions prior to 9.4.
NOLARGEFILES	<p>Creates the dump files without exceeding 2Gb. If the dump file is larger than 2Gb, then partitions will be created.</p> <p>Use the NOLARGEFILES keyword with the SIZES keyword to restrict the file sizes to a specify size less than 2Gb. Do not use NOLARGEFILES with a SIZES value that is greater than 2Gb.</p> <p>You might want to restrict file sizes to 2Gb for any of these reasons:</p> <p>You might want to move the files to another machine or operating system that does not support large files.</p> <p>The dump size is too big to fit on a single disk (or the free space on a single disk) and must be split over multiple disks.</p>
UPDATE	Use this keyword to create a file that contains only data values without dimension definitions. You can then load the file into a database that already contains the required dimensions and members.
IDENTICAL	Use this keyword to create a file that is suitable for loading into a database with exactly the same structure as your database. The time series are saved to the dump file using a numeric key, which is more efficient for loading than the default text identification.
DUMMY	Replaces all nonmissing values with 1. This function is useful if you do not want to dump your actual data.
STATUS <procedure>	Creates a procedure, with the name specified by <procedure>, that you can use to recreate the current status settings.
PRUNE	Includes only the selected dimension members in the file instead of all dimension members. The member names are keyed as literal strings, not as numeric keys. You cannot use PRUNE if you have rollups that do not sum.
PERIOD <range>	Specify PERIOD to perform a snapshot of only the data values in range specified by <range>. For example, PERIOD 05-06 dumps data for the years 2005 through 2006.

EXPAND	Expands the number of dimensions a variable can have. Note: A dump file created by SNAPSHOT EXPAND must be loaded into a database that already has its measures and dimensions created.
REDUCE	Eliminates any dimensions that have only one member selected.
OVERWRITE	Overwrites a system file with the same name.
DRILLTHRU	In addition to the time series stored in Application Server, writes selected series from Hybrid OLAP schema tables into the snap file. When you subsequently load the data from the file, all data, regardless of its origin, is loaded into Application Server as non-drillthru data.
NOATTRIBUTE	(Optional) Ensures that no attributes or attribute data are included in the dump file.

Remarks

If you select only output members from your dimension and then enter the SNAPSHOT PRUNE command, the LEVEL construct in the resulting dimension will still contain the input level.

Note: If an entire structural dimension is selected, the SNAPSHOT will include all related attribute dimensions.

Example

...This example selects Shampoo Sales and creates the dump file SHAMSALE partitioned

...into multiple files, each 250K, overwriting any existing files with the same name

```
SELECT Product Shampoo
```

```
SELECT Sales
```

```
SNAPSHOT shamsale?? 250k OVEWRITE
```

...This example selects Shampoo Sales and creates the dump file SHAMSALE:

```
SELECT Product Shampoo
```

```
SELECT Sales
```

```
SNAPSHOT shamsale
```

...Expanding on the previous example, this example puts Deodorant Sales in the file DEODSALE. The

... dimension Product is not included in DEODSALE because of UPDATE:

```
SELECT Product Deodorant
```

```
SNAPSHOT deodsale UPDATE
```

...Assume that the command SELECT Region BELOW France selects three French cities that roll up

... into France. If you enter the following command, the snap file, FNAME, snaps the Region

...dimension, which contains just the three cities and the output member France. Other regions are

... excluded. The SNAPSHOT command is performed on the data in update format and the member

... names are keyed as literal strings instead of as numeric keys:

```
SNAPSHOT fname PRUNE
```

...This example performs the SNAPSHOT command for the data values in 1999 and 2000.

```
ACROSS TIME, DOWN PRODUCT, REGION, VARIABLES
```

```
SET PER 95-00
```

```
SEL SALES
```

```
SNAPSHOT SNAPFILE PERIOD 99-00
```

...Assume you have a prototype database with SALES dimensioned by PRODUCT, using data from

SNAPSHOT

... **CHICAGO**. This example expands **SALES** to be dimensioned by **REGION** as well,
...where **CHICAGO** is just one member of **REGION**.

USE PROTOTYPE

SELECT SALES

SNAPSHOT SALESDMP EXPAND

USE NEWDATA

SELECT REGION CHICAGO

LOAD SALESDMP

...This example reverses the **EXPAND** process, eliminating any dimensions that have only one member
... selected. The result puts the variable **SALES** with Boston's data in the file **BOSSALES**,
...but is dimensioned only by **PRODUCT**:

USE NEWDATA ... SALES HERE HAS PRODUCT AND REGION

SELECT REGION BOSTON

SELECT SALES

SNAPSHOT BOSSALES

Creating a snapshot file with multiple partitions

By default, one large dump file is created. It can exceed 2Gb up to 64-bit. If you do not want a database dump to exceed 2GB, you can split the dump across multiple files (partitions). To dump a snapshot of a database to multiple partitions, you specify a wildcard within the dump file name, for example **SNAP mysnap??**. You can also specify **NOLARGEFILES** to make sure that each partition is less than 2Gb. You can also specify the **SIZES** keyword to specify the exact sizes of each partition. Application Server replaces the wildcard characters with partition numbers, for example **mysnap01**, **mysnap02**, and so on. For example:

```
SNAPSHOT mysnap?? NOLARGEFILES
SNAPSHOT mysnap?? SIZE 1G
SNAPSHOT mysnap?? SIZE 5G
```

Location of snapshot files in Windows

In Windows, snapshot files are placed in the Home directory. However, you can specify alternative locations by editing the **LSSERVER.INI** file before executing the **SNAPSHOT** command:

```
[Windows]
mysnap##=pathname\mysnap##
```

Substitute **snap##** with the name of a snapshot file partition, and substitute **pathname** with the directory into which you want the snapshot file to be placed. You should create a separate entry for each partition, for example:

```
[Windows]
mysnap01=c:\db\mysnap01
mysnap02=d:\db\mysnap02
mysnap03=e:\db\mysnap03
```

Location of snapshot files in UNIX

In UNIX client/server, snapshot files are placed in the directory defined by the **LSSHOM** environment variable. To specify a different location for the files, add environment variables to **lsstcp.sh**, for example:

```
mysnap##=pathname/mysnap##
export mysnap##
```

Substitute mysnap## with the name of a snapshot file partition, and substitute pathname with the directory into which you want the snapshot file to be placed. You should create a separate entry for each partition, for example:

```
mysnap01=/db1/mysnap01
mysnap02=/db2/mysnap02
mysnap03=/db2/mysnap03
export mysnap01
export mysnap02
export mysnap03
```

6.202 SOLVE-ENDSOLVE

Use

SOLVE-ENDSOLVE returns the simultaneous solution of two or more equations; it recalculates until it finds a value that satisfies all equations.

Syntax

```
SOLVE
      .
      . simultaneous equations
      .
ENDSOLVE
```

Example: SOLVE-ENDSOLVE

...In this example, Application Server determines the value of Commission in an iterative process by
... trying a value of profit and calculating the commission. Application Server then calculates a new
... value of Profit, recalculates the commission, and so on, until it finds a value for
...commission that satisfies both equations.

SOLVE

Commission = 0.03 * Profit

Profit = 'Gross Profit' - Commission

ENDSOLVE

6.203 SQRT()

Use

SQRT returns the square root of a number.

Syntax

CALCULATE <result> = SQRT(<input>)

Parameter	Description
<result>	Name of the result variable.
<input>	Name of the variable whose square root you want to calculate.

6.204 STATUS

Use

STATUS provides general information about the current Application Server session.

SUBTOTAL-ENDSUBTOTAL (Report)

Syntax

STATUS

Example: STATUS

This example shows the current Application Server status:

STATUS

...The output of this command would be:

Across List: # Selected

COUNTRY 3 US, UK, GERMANY

TIME

Down List:

PRODUCT 7 TV, RADIO, KITCHEN, OUTDOOR...

VARIABLE 4 SALESVOL, REVENUE, MARGIN, INCREASE

Period 98/1/1 - 98/12/31

Attached Databases: CORPDB, MKTG

Current Logic CONTRIB Report QUARTERLY

Retail Fiscal Year Starts in January, on Sunday

6.205 SUBTOTAL-ENDSUBTOTAL (Report)

Use

SUBTOTAL-ENDSUBTOTAL is a Report command that produces a subtotal of all rows included within a block. You can nest subtotal blocks.

Syntax

SUBTOTAL

 .<statement> [EXCLUDE]

 ENDSUBTOTAL <subtotal> ['<text>'] [SUPPRESS]

Parameter	Description
<statement>	One or more report statements, for example ROWS.
EXCLUDE	Omits a specified row from the subtotal.
<subtotal>	Name of the subtotal. You can use this name in expressions as if it were a TEMP variable.
<text>	Text to use in place of the name in the report.
SUPPRESS	Suppresses any column calculations that are in effect when Application Server displays the subtotal row.

Example

...This example illustrates the subtotalling of costs:

SUBTOTAL

 ROW Variable_Cost

 ROW Fixed_Cost UDATA

ENDSUBTOTAL Total_Cost

...It produces output similar to:

	1996	1997	1998
Variable Cost	13	17	15
Fixed Cost	3	4	4
	==	==	==
Total Cost	16	21	19

6.206 SUBTOTAL INCLUDE (Report)

Use

SUBTOTAL INCLUDE is a Report command that produces a subtotal of the specified rows.

Syntax

SUBTOTAL INCLUDE <row> [, ..., <row>] <name> ['<text>'] [SUPPRESS]

Parameter	Description
<row>	Name of a row. This can include names in the last down dimension.
<name>	Assigned row name. This name can appear in subsequent expressions.
<text>	Row label to use in place of the name.
SUPPRESS	Suppresses column calculations in the subtotal row.

Example: SUBTOTAL INCLUDE

...This example illustrates the subtotalling of costs:

ROW Variable_Cost

ROW Fixed_Cost UDATA

SUBTOTAL INCLUDE Variable_Cost, Fixed_Cost Total_Cost

...It produces the same results as the example in SUBTOTAL-ENDSUBTOTAL:

	1996	1997	1998
Variable Cost	13	17	15
Fixed Cost	3	4	4
	==	==	==
Total Cost	16	21	19

6.207 SUBTRACT DATABASE (Supervisor)

Use

SUBTRACT DATABASE is a Supervisor command that removes a database entry from MASTERDB, but does not remove the database file or files.

Syntax

SUBTRACT DATABASE <name>

Parameter	Description
<name>	The database name.

Example: SUBTRACT DATABASE (Supervisor)

...Remove the database entry DEMO from MASTERDB without removing the database file or files:

SUPERVISOR

SUM (Dimension)

SUBTRACT DATABASE DEMO

6.208 SUM (Dimension)

Use

SUM is a Dimension formula that defines how a group of input or output members roll up into a higher output or result member. You use this formula in a consolidation statement, which is the last statement in a dimension.

Syntax

```
<output> = { SUM [-]<member>, ... [-]<member> }
          { <member1> SUM <membern> }
```

Parameter	Description
<output>	Name of the output member or result member whose rule of consolidation is being defined.
-	Sums the difference between members when the consolidation statement involves the variance <i>between</i> members. For example, the TYPE dimension has two input members, actual and budget, whose output member is the variance between the two input members. The consolidation statement might be: variance = SUM budget, -actual.
<member>	Application Server adds the names of noncontiguous members.
<member1>	First member in a contiguous list to be added.
<membern>	Last member in a contiguous list to be added.

Example

...This example defines the Variance output member for a TYPE dimension:

INPUT

Actual, Budget

OUTPUT

Variance

variance = SUM actual, SYMBOL 45 "Symbol" 9budget

...Sum every member from zn1 through zn5 in the order in which they are defined in the dimension:

zone = zn1 SUM zn5

...This example sums specific, noncontiguous members:

footwear = SUM sneakers, sandals, pumps, slippers

6.209 SUMMARY-ENDSUMMARY (Report)

Use

SUMMARY-ENDSUMMARY is a Report block that is executed at the end of a nested level. It is only valid for nested level reports, that is, reports using STYLE LEVELTOTAL in SET-ENDSET. Two functions are provided — they are valid only within the SUMMARY-ENDSUMMARY block:

Function	Description
NLEVELPRINTED()	For the current nested level, show totals and counts for printed items following cut-off criteria.
<PrintedCount> - (Optional)	Can be used with NLEVELPRINTED() to show the count of printed items. For example:
ROWS NLEVELPRINTED() {"Total Printed", <PrintedCount>}	

NLEVELOTHER() For the current nested level, show totals and counts for non-printed items following cut-off criteria.

<OthersCount> - (Optional) Can be used with NLEVELOTHER() to show the count of non-printed items. For example:

ROWS NLEVELOTHER() {"Total Other", <OthersCount>}

6.210 SUPERVISOR

Use

SUPERVISOR starts the Supervisor subsystem, where you can manage database and user information in MASTERDB.

Syntax

SUPERVISOR

Example: SUPERVISOR

...This example removes the SALES database from MASTERDB:

SUPERVISOR

REMOVE DATABASE sales

...You can also enter SUPERVISOR without going into and getting out of the Supervisor subsystem:

SUPERVISOR REMOVE DATABASE sales

6.211 SWITCH

Use

Use the SWITCH command to log into Application Server again as a different user. Or, you can also switch to a different Work database in the same session while saving the previous Work database and staying logged in as the same user.

Syntax

SWITCH { <username> [PASSWORD <password>] | WORKDB [<name>] }

Parameter	Description
<username>	The user name you want to log in as.
PASSWORD	Indicates that the user name has a password.
<password>	The password for that user.
WORKDB	Saves the prior Work database in the session and creates a new Work database named DBxxxxx while remaining logged in as the same user. At a later point, you can switch back to the previous Work database and start up again with your previous status. Note: See the CSWITCH command if you want to switch to a different Work database and delete the previous Work database.
<name>	Saves the prior Work databases in the session, and switches to the Work database name specified while remaining logged in as the same user. If the name already exists, it is used and the state is restored. If that Work database exists but is not a valid Work database, you get an error and rollback. If the Work database does not exist, a new one is created with the blocksize and blocks defined for the current user, that is, just as original Work database would have been created. Note: Some settings like the ACROSS and DOWN commands and SELECT statements are saved in a Work database. If you switch back to a previous Work database, you will get those settings, and as a result, your view may change.

SYNC

Tip: The WORKDB feature is helpful in applications that use User-Defined Hierarchies, such as the strategy management application. The application typically creates and clears User-Defined Hierarchies for each transaction. By switching between multiple Work databases, you can maintain the appropriate state and avoid the constant creating and clearing of the User-Defined Hierarchies.

6.212 SYNC

Use

SYNC specifies how the operating system file cache should perform physical disk writes when updating a database with modified data blocks.

Normally at the end of any Application Server command (if you haven't done an explicit CHECKPOINT FREEZE) or at a CHECKPOINT UPDATE, Application Server will write modified database blocks to the operating system. To ensure database integrity, Application Server then also asks the operating system to physically write those blocks to disk and waits for the operating system to confirm that it has done the physical write to disk. Only then does Application Server complete the transaction. The only benefit to this process is if there is a power outage or operating system crash while the physical write is in progress, upon a restart, Application Server can rollback to the last complete transaction and ignore the failed write. In this way, the integrity of the database is guaranteed. The drawback to this method is that physical i/o is sometimes slow and Application Server waits for it to complete, so this slows down many Application Server commands. This process prevents the operating system from performing lazy physical writes at a more optimal time — that is, when it needs to or when there is idle time on the machine.

If you set SYNC OFF, Application Server does not ask the operating system to force modified blocks to write to disk at the end of a command or CHECKPOINT, and does not wait for any physical i/o. This often improves performance.

It is recommended to use SYNC with the OFF keyword as the first command in an Application Server session that includes commands that modify the database such as CONSOLIDATE, CALCULATE, REMOVE, and CREATE. For example, you may want to put SYNC OFF as the first command in an AUTOUSE procedure. The SYNC OFF command allows the o/s file cache to do lazy physical disk writes. If SYNC OFF is set, all subsequent Application Server requests to the o/s for writes allow the o/s to handle physical writes and Application Server proceeds without waiting for its completion. Application Server never waits for physical i/o during the whole session. You benefit from performing lazy physical disk writes because it greatly shortens the time it takes to execute the commands. The drawback to this is that if there is a power outage or a machine crash and the operating system has not completed all of its physical i/o, then upon a restart, an Application Server database may be corrupted.

When using SYNC OFF, once all the commands that modify the database are complete, you can then use SYNC ON to allow Application Server to handle the writing of database blocks to disk. When SYNC is ON, at the end of every Application Server command, the sync will occur. SYNC ON is the default setting.

Syntax

SYNC { OFF | ON | NOW | SHOW }

Parameter	Description
OFF	Allows the OS to do lazy physical i/o.
ON	Allows Application Server to handle the writing of database blocks to disk. This is the default setting. When SYNC is ON, at the end of every Application Server command, the sync will occur.
NOW	Forces a disk i/o now.
SHOW	Shows whether SYNC ON or SYNC OFF is the current setting.

6.213 SYNONYM

Use

SYNONYM starts the Synonym editor, where you can define synonym sets by using Edit Syntax 1. You use these sets to convert external source file field values to internal Application Server dimension member names when loading data. You reference these sets with the DEFINE SYNONYM command.

Alternatively, you can use Edit Syntax 2 to create a synonym set to create alternate long names for dimension members — for use when you want any Application Server displays to report different member long names under different circumstances. For example, you might want a report to appear in English for one group of users, but then have it appear in Spanish for a different group. You reference these sets with the SET LABEL command.

Syntax

SYNONYM [<synonym>] [;<database> | ;EXTERNAL|LOCAL]

Edit Syntax 1

<internal> = <external> [..., <external>]

<internal> = BLANK

Edit Syntax 2

[NOCASECONVERT]

<alternative_label> = <internal>

Parameter	Description
<synonym>	The name of a new or existing synonym set up to 96 bytes. If you do not specify a name, Application Server uses the default synonym set if it is defined, or the last <synonym> set you edited.
<database>	Name of the database where the synonym set is located. If <i>you</i> do not specify a database, Application Server uses your Use database.
EXTERNAL	Indicates the synonym set is a text file that is not in an Application Server database. If the synonym set is a DOS file, its name cannot have an extension.
LOCAL	Indicates that the text file is a client <text> file.

Note: You can create or edit a synonym set with the EXTERNAL or LOCAL extension, but you must copy it into a model and compile it before it can be referenced in a DEFINE SYNONYM statement.

Editor Options

Edit Syntax 1

<internal>	An Application Server short name.
<external>	A name in an external source.
BLANK	Indicates that the value in the external source is null or missing.

Edit Syntax 2

[NOCASECONVERT]	Used when creating a synonym set for alternate dimension member long names to be used with the SET LABEL command, NOCASECONVERT prevents Application Server from converting labels to uppercase when the set is compiled.
<alternative_label>=<internal>	Defines the alternate long name to use for the specified internal short name. Note that you must specify the names in the opposite order that you would use for data loads — specify the alternative member name first, and then the internal short name.

TABLE()

For example, in the sample JUICE database, where the CHANNEL dimension has members with short names C1,C2,C3, the following synonym set is valid:

NOCASECONVERT

'MyDirectSynonym' = C1

'MyDistributorsSynonym' = C2

'MyTotalSynonym' = TOTAL_CHANNEL

Example: SYNONYM

...This example creates a synonym set called Region_Names for the Region dimension. The dimension

... member names USA, UK, and CANADA are represented in the source file as C101, C102, and

... C103. The last line indicates that missing entries in the external

... source should be assigned to the dimension member 'N/A'.

SYNONYM Region_Names

USA = C101

UK = C102

CANADA = C103

'N/A' = BLANK

6.214 TABLE()

Use

TABLE retrieves a value from a table of numbers, based on an index.

Syntax

CALCULATE <result> = TABLE(<value>, <c1>, <c2>, ..., <cn>)

Parameter	Description
<result>	Name of the result variable.
<value>	Index value between 1 and <i>n</i> , where <i>n</i> is the number of values in the table. If the value of <i>n</i> is outside the table, Application Server returns a missing value.
<c1>, ..., <cn>	Table of numbers.

Example

...In this example, Country has the values 1, 2, or 3. Tax_Rate has the value of .32 for

...country 1, .45 for country 2, and .27 for country 3.

CALCULATE Tax_Rate = TABLE(Country, .32, .45, .27)

6.215 TABS (Report)

Use

TABS is a Report command that inserts a tab character between columns. You use TABS in a SET-ENDSET construct.

Note: When using the DISPLAY command to display a report with tabbed columns, you can specify an alternate delimiter, such as a comma, for easy transfer into Microsoft Excel.

Syntax

TABS

6.216 TABS SINGLE (Report)

Use

When using the DISPLAY command to display a report with tabbed columns, you can specify an alternate delimiter, such as a comma, for easy transfer into Microsoft Excel. If you do not override the tab character in the DISPLAY command, two tab characters are inserted in front of the long name. TABS SINGLE suppresses one of these tab characters, so that the long name is preceded by a single tab character.

TABS SINGLE is a Report command — you use TABS SINGLE in a SET-ENDSET construct.

Syntax

TABS SINGLE

Example

This example inserts a tab character between columns:

SET

PREFACE '123'

TABS

COLUMN ONCE

ORDER 0-12

ENDSET

6.217 TEXT (Report)

Use

TEXT is a Report command that displays text over the specified columns.

Syntax

```
TEXT {LEFT} [<columns>] [ '<text>' ] [<keyword>]
    {CENTER}
    {RIGHT}
    {OVER [1-N]}
    {<n>}
```

Parameter	Description
LEFT	Left-aligns the text.
CENTER	Center-aligns the text.
RIGHT	Right-aligns the text.
OVER	Centers the text over the specified columns.
1-N	Centers text based on the actual number of columns. <N> is a symbolic column number that causes titles and text to be centered over the actual width, instead of over the stated width.
<n>	A positive integer indicating the number of spaces to indent text from the left margin.
<columns>	One or more column numbers over which you want the text to appear. You can specify either a range of columns separated by a dash (-), such as 3 - 5, or a range of columns and an increment such as 1 - 7 - 2, indicating columns 1 through 7 incremented by 2, or columns 1, 3, 5, and 7.

TIME

Note: A column number refers to the columns as they appear on the report page. With ORDER in effect, the column numbers are determined by the current ACROSS/DOWN and SELECT commands.

<text> Text you want to display, enclosed in single quotation marks (' ').

<keyword> One of these keywords whose value you want to display, enclosed in angle brackets: clock, date, latest, page, heading, or dimension, where *dimension* is the name of a down dimension (but not the last down dimension).

Example

...Print Page No. followed by the current page number of the report. For example, Page No. 5.

TEXT 'Page No. ',<page>

...Print Country of Origin followed by the current country name as in Country of Origin is Canada.

TEXT 'Country of Origin is ',<country>

...This example puts the default column headers between lines of equal signs:

COLHEADING

TEXT '====='

TEXT <heading 1>

TEXT '====='

ENDCOLHEADING

...If the column headers are months, the resulting column headings appear as:

JAN 98 FEB 98

6.218 TIME

Use

TIME starts the Application Server editor, where you can create or edit a time set used in lists and reports.

Note: If you manually create a time set, the last OUTPUT or RESULT member must have a label.

Syntax

TIME [<time>] [;<database> | ;EXTERNAL|LOCAL]

Editor Syntax

[VIRTUALCONVERT {BEFORE| AFTER}]
INPUT [TEMP] <periodicity> <period> [- <period>] [AS | NAMED <inputs>]

Parameter	Description
<time>	Name of a new or existing time set up to 96 bytes. If you do not specify a name, Application Server uses the default time set if it exists, or the last time set you edited.
<database>	Name of the database where the time set is located. If you do not specify a database, Application Server uses your Use database.
VIRTUALCONVERT	Changes the order of calculations when a virtual variable is used in conjunction with a time set with output calculations. Use BEFORE when you are summing ratios or percents and you want the time set calculation to happen first, before the virtual variable calculation. Use AFTER when you are doing a % change calculation on these same virtual variables.

Notes: The SET VARIABLE command also allows you to use a CONVERT BEFORE or AFTER statement. The SET VARIABLE command requires exclusive access to the dimensional model. Use the VIRTUALCONVERT when you do not have exclusive access.

If a VIRTUALCONVERT statement exists in the time set, it overrides any SET VARIABLE CONVERT settings.

EXTERNAL Indicates the time set is a text file that is not in an Application Server database. If the time set is a DOS file, its name cannot have an extension.

LOCAL Indicates that the time set is a client text file.

Note: You can create or edit a time set with the EXTERNAL or LOCAL extension, but you must copy it into a model and compile it before it can be used in the model.

Example

...Assume you have the following time dimension named Comp:

input month latest as t1

input month latest minus 12 as t2

output advance, progress

advance = t2 - t1

progress = t2 / t1

...Assume you also have a Product dimension with members Heavy, Cars, and Trucks, a Class

... dimension with a member Actual, and a logic set that calculates a variable Contrib.

...When you display the data with the following statements:

ACROSS Class, Variables, Product DOWN Comp

SET LATEST 98/2

LIST

...The output appears as:

	Actual		
	Contrib		
	Heavy	Cars	Trucks
Feb 1999	744.30	188782.92	147419.01
Feb 1998	591.35	168789.21	145052.61
Advance	152.95	19993.71	2366.40
Progress	1.25	1.11	1.01

...Application Server lists the four members of Comp down the page. The first row of variables in the ... list corresponds to the first dimension member: the latest month from the chosen period, or ... February 1999. The second row of variables corresponds to the second dimension member: the ... latest month minus twelve months, or February 1998. The third and fourth rows correspond to the ... calculated output dimension members Advance and Progress.

6.219 TITLE (Report)

Use

TITLE is a Report command that displays the specified text as a centered report title.

TOTAL()

Syntax

TITLE '<text>'

Parameter	Description
<text>	Text you want to print as a title.

Example

...This example prints 1998 Revenue as the report title:

TITLE '1998 Revenue'

6.220 TOTAL()

Use

TOTAL returns the accumulated values over all the selected time periods.

Syntax

CALCULATE <result> = TOTAL(<input>)

Parameter	Description
<result>	Name of the result variable.
<input>	Name of the variable to total.

Example: TOTAL

...This example totals all the values of Cost over the selected period:

CALCULATE Total_Spent = TOTAL(Cost)

6.221 TRACE

Use

TRACE sends a copy of commands or command output, or both, to a specified destination. The default is to overwrite the exporting file, set, or symbolic name set.

Syntax

TRACE [COMMANDS | TIMING | OUTPUT | BOTH] {<destination> [APPEND]} OFF | UPDATE}

Parameter	Description
COMMANDS	Copies either commands from the command window or any job to a printer or destination.
TIMING	Specifies the time it takes to execute the commands from the command window or any job to a printer or destination. In the output, the timing is displayed first, and then the command. Notes: <ul style="list-style-type: none"> If you specify TRACE TIMING TERMINAL, timing information appears after the command output. SHOW SETTINGS doesn't show TRACE TIMING settings.
OUTPUT	Copies output (data, error messages, reassurance messages) to the destination.
BOTH	Copies both commands and output to the destination (default).
<destination>	One of the following destinations: TERMINAL - Your terminal. PRINTER - The default printer on your system.

<setname> [;<database> | ;EXTERNAL] , where:

<setname> - Name of a set. If you copy commands, Application Server creates a procedure. If you copy output or both commands and output, Application Server creates a document.

<database> - Name of the database where the set is located. If you do not specify a database, Application Server uses your Use database.

EXTERNAL - Indicates the set is a text file that is not in an Application Server database. If the set is a DOS file, its name cannot have an extension.

Note: When specifying a path and filename for the destination on UNIX, make sure that you do not exceed 53 bytes.

OFF	Turns off a TRACE command. You must enter TRACE OFF to end a trace. Note: You can have both TRACE COMMANDS and TRACE BOTH active at one time. For example, you can enter a command in the form TRACE COMMANDS <i>testproc</i> to capture only the commands you enter in the procedure TESTPROC, and a command in the form TRACE BOTH <i>tracecheck</i> to capture both commands and output in the document TRACECHECK. To turn off these commands, you need to enter TRACE COMMANDS OFF and TRACE BOTH OFF, respectively.
APPEND	Overrides the default and adds the trace to the end of an exporting file or set, without overwriting it.
UPDATE	When used, each command and its output is sent to the destination when it is executed instead of being buffered.

Example: TRACE

...This example sends everything that appears on your

...screen between the two TRACE commands to the printer:

TRACE BOTH PRINTER

.

. (all dialogue on the screen here is sent to the printer)

.

TRACE BOTH OFF

...This example sends all commands you enter, and all commands from any procedures you enter with

... JOB, to the procedure NEWPROC, until you enter TRACE BOTH OFF:

TRACE COMMANDS Newproc

. (all subsequent commands are copied to the procedure newproc)

. (issue a series of Application Server commands)

TRACE COMMANDS OFF

...This example dumps the generated SQL to a document set:

TRACE ON document_set

TRACE OFF

...This example directs the TRACE output to a UNIX file /lsserver/this_is_a_trace_file:

TRACE /lsserver/this_is_a_trace_file

...This example sends everything that appears on your screen to a file called "t_file" which is not an

... Application Server database. It sends each command and its output to the destination when it is

...executed instead of being buffered.

TRACE BOTH t_file;EXTERNAL UPDATE

TYPE

6.222 TYPE

Use

TYPE displays the contents of a set.

Syntax

TYPE

```
{ [<settype>] <setname> [TOP integer] }
{ LOGIC %<vvariable> }
```

Parameter	Description
<settype>	Type of set: dimension, document, logic, procedure, report, synonym, or time. If you do not specify a set type, Application Server displays the first set it finds with the specified name.
<setname>	Name of the set you want to display.
TOP <integer>	The first number lines of the set are typed (counting from the top). The integer must be greater than 0.
LOGIC %<vvariable>	Types a virtual variable's hidden logic set. For example, TYPE LOGIC %margin.

Example

...This example displays the dimension Country:

TYPE dimension Country

INPUT usa, canada, germany, england, france

OUTPUT 'n america', europe

RESULT world

'n america' = usa + canada

europe = england + france + germany

world = 'n america' + europe

6.223 UDASH (Report)

Use

UDASH is a Report command that prints a line of dashes or a specified character across a page.

Syntax

UDASH ['<t>']

Parameter	Description
<t>	Character to print in a line. If you do not specify a character, Application Server displays a line of dashes.

Example

...This example prints -----:

UDASH

...This example prints *****:

UDASH '*'

...This example prints =====:

UDASH '='

6.224 UDATA (Report)

Use

UDATA is a Report command that underlines the data in the previous row with dashes or a specified character.

Syntax

UDATA ['< t> ']

Parameter	Description
<t>	Character with which to underline data. If you do not specify a character, Application Server displays a line of dashes. Note: If you use the underline option in a SET-ENDSET block, you can specify the width of the underlining for each column.

Example

...This example underlines data with -----:

UDATA

...This example underlines data with *****:

UDATA '*'

6.225 UNAME (Report)

Use

UNAME is a Report command that underlines the variable name in the previous row with dashes (-) or a specified character.

Syntax

UNAME ['<t> ']

Parameter	Description
<t>	Character with which to underline variable names. If you do not specify a character, Application Server displays a line of dashes.

Example

...This example underlines the variable name with -----:

UNAME

...This example underlines the variable name with +++++++:

UNAME '+'

6.226 USE

Use

USE specifies the database to use.

Syntax

USE <database> [EXCLUSIVE | SHARED | READ] [RETAIN] [NOVARDB]

Parameter	Description
<database>	Name of the database you want to use.

USE (Access)

EXCLUSIVE	Indicates that only you can read and update the database (default).
SHARED	Indicates that all users can read and update unique areas of the database. Users can change time sets, report sets, documents, and procedures within the database. Users cannot edit data that is selected by another user; each area can be edited only by a single user at a time. Users cannot change the structure of the database by editing dimensions sets.
READ	Indicates that all users can read, but not update, the database.
RETAIN	Does not detach any databases you specified in a previous USE command.
NOVARDB	Does not attach any variable databases (by default, all the variable databases are attached with the same access as specified by the USE command).

6.227 USE (Access)

Use

USE is an Access command that specifies the external data file you want to access.

Syntax

USE { <file> | <document>; <database> } [OVERWRITE] [OEM]

Parameter	Description
<file>	Name of a file or path up to 64 alphanumeric characters. If you specify a path name, you must enclose it in single quotation marks (' ').
<document>	Name of an Application Server document up to 24 alphanumeric characters.
<database>	Name of the database where the Application Server document is located.
OVERWRITE	Overwrites a file or document with the same name.
OEM	Indicates the file contains national alpha characters generated under DOS that need to be translated to the Windows ANSI equivalents.

Example: USE

...Here, FINANCE is the current Use database:

SHOW DATABASE

...The output of this command would be:

USE Database: FINANCE Maximum Observations: 200

	Access	Maxblocks	% Available	Mode
TESTDB	READ	500	41%	READONLY
FINANCE	UPDATE	200	49%	EXCLUSIVE
MARKET	UPDATE	500	64%	SHARED

...The following statements change the database to MARKET:

USE Market

SHOW DATABASE

...The output of this command would be:

USE Database: ACCOUNT Maximum Observations: 400

	Access	Maxblocks	% Available	Mode
TESTDB	READ	100	41%	READONLY
FINANCE	UPDATE	200	49%	EXCLUSIVE
MARKET	UPDATE	500	64%	SHARED

6.228 UTEXT (Report)

Use

UTEXT is a Report command that underlines all text in the previous line with dashes or a specified character.

Syntax

UTEXT [' <t> ']

Parameter	Description
<t>	Character with which to underline text. If you do not specify a character, Application Server displays a line of dashes.

Example

...This example underlines text with -----:

UTEXT

...This example underlines text with =====:

UTEXT '='

6.229 VALUE()

Use

VALUE returns the value of the specified observation for a variable.

Syntax

CALCULATE <result> = VALUE(<variable>, <n>)

Parameter	Description
<result>	Name of the result variable.
<variable>	Name of the variable for which you want the observation. Variable names that use special characters should be in single quotation marks (' ').
<n>	Number of the time series observation in the variable. The number 1 returns the first observation; NCASES() returns the last.

Example

...Suppose you have a time dimension Quarter with the following definition:

INPUT month 1 – 3

INPUT quarter 1

...and a report with the following definition:

ROWS Sales

ROWS Sales / VALUE(Sales, 4) 'Mon/Qua'

...When you display the report using ACROSS Quarter DOWN

...Variables, the following appears. Here, VALUE(Sales, 4) returns 60.

	Jan	Feb	Mar	Qua 1
Sales	10.00	20.00	30.00	60.00
Mon/Qua	0.16	0.33	0.50	1.00

6.230 VERSION

Use

VERSION displays the current Application Server version number.

Syntax

VERSION

Example: VERSION

...This example displays the Application Server version number:

VERSION

...The output of this command would be:

Application Server (TM)

Version 9.x.0 for Windows

Copyright (C) SAP AG 200x

Reference 6010 on 03-01-2008 10:13:37

This software program is licensed by SAP AG for
use pursuant to the terms and conditions of a license agreement.

6.231 WHEN-ENDWHEN

Use

WHEN-ENDWHEN executes a block of statements based on a specified condition.

Syntax

```
WHEN <condition>
    .
    .      <statements>
    .
[ELSEWHEN <condition>
    .
    .      <statements>]
.
[ELSE
    .
    .      <statements> ]
.
ENDWHEN
```

Parameter	Description
WHEN	Indicates the start of a block. If the condition is true, executes the statements immediately following the condition. If the condition is false, proceeds to the next conditional block, if it exists.
<condition>	A logical condition using any arithmetic or logical operators.
<statements>	Any sequence of statements or commands.
ELSEWHEN	If the condition is true, executes the statements immediately following the condition. If the condition is false, proceeds to the next conditional block, if it exists.
Note: There is no limit to the number of ELSEWHEN blocks.	

ELSE	If the condition is true, executes the statements immediately following the condition. If the condition is false, exits from the loop.
ENDWHEN	Indicates the end of a block.

Example

...Assume that the UK has a tax rate of 30%, France has a tax rate of 28%,

...and all other countries have a rate of 32%. You can calculate the tax rates as follows:

WHEN Country EQ UK

Tax Rate = .3

ELSEWHEN Country EQ France

Tax Rate = .28

ELSE

Tax Rate = .32

ENDWHEN

...Assume that a company offers a discount that is 5% for volume over 250 units, 10% for volume over 500 units, and 15% for volume over 1000 units. Using the conditional operator LE, this example

... shows how to calculate the discount:

WHEN Volume LE 250

Discount = 0.0

ELSEWHEN Volume LE 500

Discount = 0.05

ELSEWHEN Volume LE 1000

Discount = 0.1

ELSE

Discount = 0.15

ENDWHEN

6.232 WHILE-ENDWHILE

Use

WHILE-ENDWHILE executes a block of statements until a specified condition becomes false.

Syntax

```
WHILE <condition>
    .
    .      <statements>
    .
ENDWHILE
```

Parameter	Description
WHILE	Indicates the start of a loop. While the condition is true, executes the statements in the loop. When the condition is false, exits from the loop.
<condition>	A logical condition using any arithmetic and logical operators.
<statements>	Any sequence of statements or commands.
ENDWHILE	Indicates the end of a loop.

6.233 WRITE (Access)

Use

WRITE is an Access command that writes data to an external file. It is possible to write files that exceed 2Gb.

Use WRITE in ACCESS EXTERNAL to write data to text files to be used as load files in other tools. For example, you can write data to a text file and use it as a data source to load into SAP BW.

The output is written to the file you specify in the USE statement, according to the layout you specify in the DESCRIPTION statement.

Syntax

```
WRITE {APPEND}  
      {NEW}  
      {UPDATE}  
      {EXISTING}  
      {OVERWRITE}
```

Parameter	Description
APPEND	Appends records to the end of a file.
NEW	Writes only new records to a file.
UPDATE	Writes new records to a file and updates any records that already exist in the file.
EXISTING	Updates records that already exist in a file.
OVERWRITE	Empties the file before writing all records to it.

Note:

If you have the Application Server database set to display the short names with SET SHORT, Application Server sends the short names for the variables and dimensions to the file. If you set the database to the display labels with SET LONG, Application Server sends the labels to the file.

Example: WRITE

...Write all records to the external file specified with USE overwriting the file's contents if not empty:

USE Shipment

WRITE OVERWRITE

6.234 XRAY (Supervisor)

Use

XRAY is a Supervisor command that validates records in the database.

Syntax

```
XRAY <name> [RECORDS] [NOTREE]
```

Parameter	Description
<name>	The database name.
RECORDS	This option validates all the records in the database. This option is extremely time consuming for large databases and you should use it with care. If you do not use the RECORDS option, Application Server does not check each record, and runs much faster, but provides somewhat less validity checking.
NOTREE	Switches off the checking of B-Trees. Use this option to do a brief check on a large database.

Example: XRAY (Supervisor)

...In this example, Application Server validates every record in the MYDB database:

XRAY MYDB RECORDS

...In this example, a quick check of the MYDB database is performed. B-Trees are not checked.

XRAY MYDB NOTREE

6.235 YIELD()

Use

YIELD returns a scalar value containing the yield of a bond or security with a given interest rate.

Syntax

YIELD(<value>, <rate>, <periodicity>, <maturity>, <buyprice>, <buydate>, <sellprice>, <selldate>)

Parameter	Description
<value>	Scalar amount of the face value of the bond or security.
<rate>	Scalar annual rate or time-series rate.
<periodicity>	Periodicity of the dividend payment: yearly, quarterly, or monthly.
<maturity>	Date when the bond or security matures.
<buyprice>	Scalar purchase price.
<buydate>	Date when the bond or security was bought.
<sellprice>	Scalar price at which the bond or security is sold. This is the same as the face value if the bond is held to maturity.
<selldate>	Date when the bond or security is sold or matures.

7 Working with Hybrid OLAP

7.1 What Is a Schema?

A schema is a set of relational database tables that are organized to make multidimensional analysis easier to perform. The Hybrid OLAP analysis environment is a combination of an Application Server multidimensional model with a relational schema.

You can create a schema from the Application Server IDQL command line. When the schema is in place, Application Server generates the SQL code required to retrieve, cache, and update the schema without interaction from the user. (The SCHEMA subsystem SPY command allows you to view the SQL code that Application Server generates.)

All components of an Application Server multidimensional model, including dimensions, variables, attributes, labels, and data, are represented in the schema. For example, Application Server dimensions are defined in the Dimensions table in the schema.

All of the tables in a schema are identified by a schema prefix that the user sets. For example, the Dimensions table in a schema with the prefix DEMO is called DEMO_DIMENSIONS. When you create several schemas in a single database or Oracle tablespace, you can distinguish the tables that belong to each schema by their prefix.

Note: Schemas are sometimes referred to as star or snowflake schemas, because of the pattern the tables form when drawn in an entity/relationship diagram. The Hybrid OLAP schema is referred to as a galaxy schema.

7.2 How a Schema is Structured

A schema consists of relational tables representing all components of the Application Server database. The structure of the Hybrid OLAP schema is such that internal multidimensional data access can be transformed into efficient SQL. This ensures good performance when doing multidimensional data access in relational storage.

A schema is composed of the following tables:

Base tables: These contain information about the periodicities, dimensions, variables, and consolidation types that exist in the schema.

Change tables: These store updates to existing Dimension and Fact tables, allowing for faster consolidation.

Dimension tables: These describe the individual members of each dimension, their hierarchical relationships, and attribute information.

Fact tables: These store the actual time-series data for variables. Data for any given variable may be split by time across several Fact tables. Optionally, you can format fact tables to store time down in fact tables.

Format table: The Format table specifies Application Server formatting information.

Master table: The Master table contains the fiscal calendar information, the orientation of how time information is stored in the fact tables, and Application Server version information.

Reference tables: These are optional, user-created tables, one per Dimension table. They store labels for the members and variables. Using Reference tables, you can define multiple labels for a given dimension member, enabling easy support for multi-lingual labels.

Tracker table: The Tracker table is at the heart of the Hybrid OLAP schema. It keeps track of the Fact tables, which store time-series data, and controls how data for a particular variable and dimension level combination is to be consolidated.

7.3 Creating a Schema

When you create a relational schema, the components of an Application Server model, such as dimensions, variables, and data, are exported to relational tables. You need to create a relational table schema before you can use relational drill through, or exploit Hybrid OLAP's storage independence.

You can create a schema for the Hybrid OLAP analysis environment using the EXPORT commands in the Hybrid OLAP SCHEMA subsystem of Application Server at the IDQL command line

Creating a schema at the IDQL command line

You use the EXPORT commands in the Hybrid OLAP SCHEMA subsystem of Application Server to create a relational schema. To export all components of the Application Server multidimensional model in the correct order in a single step, you can use the EXPORT SCHEMA command.

Alternatively, you can issue component EXPORT commands to maintain greater control over the creation and placement of tables and indexes in the RDBMS. The component EXPORT commands feature options that allow the placement of data and indexes on different RDBMS storage segments of different disks. The component EXPORT commands must be issued in the following order to maintain relational integrity:

EXPORT BASE

EXPORT DIMENSION

EXPORT VARIABLE

EXPORT DATA

Make sure you export all of the pieces at each stage before moving on to the next one. For example, you must export all required dimensions before you export variables, and all required variables before you export any corresponding data.

You can also use the EXPORT SKELETON command to create an empty schema ready for population from other tables.

After the relational schema is in place, you can either create a new Application Server model, or adapt an existing one, to point to this schema, using the IMPORT commands or the extensions to the SET VARIABLE command.

Note: Each schema you create should have a unique prefix that serves to identify the related tables.

7.4 Creating a Hybrid OLAP Model

7.4.1 Deciding to Build a Standard or a Hybrid Model

You can build the entire dimensional model in Application Server if you want all the dimension data to be stored in the dimensional model. Application Server is optimized to allow for unlimited dimension members. For information about building a dimensional model, see the Application Server section of this help.

You may prefer to build a Hybrid model instead of a standard Application Server model for any of these reasons:

You have very large dimensions that you'd prefer to store in the RDBMS rather than in the dimensional model.

You want to use on-the-fly consolidation to configure specific level combinations and do some fine-tuning. **Note:** It is also possible to use on-the-fly consolidation in a standard model, but you can only configure based on dimension inputs and outputs.

You want the flexibility to use the data stored in the fact tables in other products or applications that support the same standard table format

7.4.2 Seven Steps to Create a Hybrid OLAP Model

To build a Hybrid OLAP model, you first build the dimensions and variables in an Application Server model and export the schema to relational tables. Then you read information into the relational tables about the Hybrid dimensions and refresh the Application Server model with the complete information about the Hybrid dimensions. Lastly, you build the tracker table, load data into the fact tables, and consolidate the data.

This section of Hybrid OLAP Help covers the general steps to build a Hybrid dimensional model.

7.4.2.1 Step 1 - Build the Dimensional Model

If you already have an existing Application Server model that you want to transform into a Hybrid model, you can skip this step and start to export the model to relational schema tables. If the model already has data loaded, you would have to export the data too.

If you do not have an existing Application Server model, the first step to create a Hybrid model is to create a model in Application Server that outlines the dimensions and variables in the model.

Procedure

```
supervisor create database salesdb blocks 200000 members 100000 observations 1000
use salesdb
```

```
access Islink
```

```
connect Base
```

```
...Build the Product dimension
```

```
select device, category, devlbl, catlbl from product.txt
```

```
construct dimension product level device, category label devlbl, catlbl
```

```
compile dimension product
```

```
...Build the Region dimension
```

```
select city, market, territory, citylbl, mktlbl, terrlbl from region.txt
```

```
construct dimension region level city, market, territory label citylbl, mktlbl, terrlbl
```

```
compile dimension region
```

```
end
```

```
...Create the variables. Account is a Hybrid dimension and Channel and Type are
...small dimensions created in steps 2 and 3
```

```
create monthly sales by account, channel, product, region, type
```

```
create monthly units by account, channel, product, region, type
```

7.4.2.2 Step 2 - Export the Model

Once you have created all the dimensions and variables in the dimensional model, you export the Application Server model to generate the Hybrid OLAP schema tables in the RDBMS. You can then use the Transformer to populate all levels of the large dimension tables with all the dimension members.

You can export in the following scenarios:

Export the base, the dimensions, and the variables individually. This gives you the chance to control many options when exporting.

Export the entire dimensional model including the base, the dimensions, and the variables all at once. This is the simpler and easier method of exporting because it takes only one command. You do not have the control over options specified above though.

With any export method you use, you can also do the following:

If the model already has data loaded in it, you can export the data to the fact tables too.

If you want to be able to use fact tables in any other product applications, you can structure the fact tables with time information stored in one column. **Note:** By default, Hybrid OLAP stores information with each time period defined as a separate column in the fact tables. **Tip:** Store time going down the table if you want to optimize the time it takes to load and consolidate data. Store time going across the table if you want to optimize end-user performance.

Procedure to export a model

Use the EXPORT SCHEMA command to create base tables in the schema, and export all dimension and variable information. For example:

..... **Export base model to create HOLAP schema tables in RDBMS.**

schema

..... **Use SPY TERMINAL to trace to the screen the SQL commands issued**

..... **by Application Server**

spy terminal

connect Holap

prefix ABC

export schema nthreads 2

spy off

end

Procedure to export a model and specify options

For example:

schema

..... **Use SPY TERMINAL to trace to the screen the SQL commands**

..... **issued by Application Server**

spy terminal

connect Holap

prefix ABC

..... **Export base model to create HOLAP schema tables**

export base nthreads 2

Note: The Table Parameters table contains columns that correspond to all of the relational database table creation parameters that can be supplied as part of an Oracle SQL CREATE TABLE or CREATE INDEX statement. You can modify these parameters to control how and where Hybrid OLAP creates the rest of the schema tables and indexes. Additionally, you can set the size for each table and index, and specify privileges for other database users on the objects you create.

EXPORT DIMENSION Product, Region UNRECOVERABLE

Creating a Hybrid OLAP Model

```
EXPORT DIMENSION Channel, Type  
EXPORT DIMENSION Account NOANALYZE DIRECTLOADPATH
```

..... **Export the variables and use short names rather than codes in the fact tables.**

```
EXPORT VARIABLE * USENAMES
```

```
spy off
```

```
end
```

Notes:

In Hybrid OLAP models, dimension names should be of no more than 17 characters.

The following tables are created when you export the model:

```
prefix_dimension1  
prefix_dimension_D  
prefix_BY_DIM  
prefix_dimension2  
prefix_dimension2_D  
prefix_CONSOLIDATIONS  
prefix_DAY_NAMES  
prefix_DIM_TYPES  
prefix_DIMENSIONS  
prefix_FMT  
prefix_LEVELS_dimension1  
prefix_LEVELS_dimension2  
prefix_MASTER  
prefix_MONTH_NAMES  
prefix_PERIOD_FORMAT  
prefix_TABLE_PARAMS
```

7.4.2.2.1 Fact tables can be used in other applications

The default schema fact tables are built with time going across. Optionally, Hybrid OLAP's open architecture allows you to store time-related information down the column in fact tables. This format is common in other applications, and allows you to use the fact tables in those applications if desired.

To create a model that can be used in other applications, you follow the same steps as though you were building a standard Hybrid OLAP model. When you are exporting the model, you include the TIME DOWN keywords in the command to specify that you want to store the time-related information down the columns in the fact tables.

You can export a dimensional model with time going down columns in two ways:

Export without specifying any aliases for dimension column names, or any data type specifications for columns. Use the EXPORT SCHEMA command with the TIME DOWN keyword for this method.

Export while specifying aliases for dimension column names and data type specifications for columns. Use the EXPORT BASE, EXPORT DIMENSIONS, and EXPORT VARIABLES commands with the TIME DOWN keyword on the EXPORT BASE command.

You include the TIME DOWN keyword on only one EXPORT command to allow the fact table configuration. All proceeding EXPORT commands assume the time down configuration.

Exporting a model with time stored down the fact table columns

This topic explains how to export a dimensional model so that time is stored down the column in fact tables. This example assumes that the model has just been created and has no data loaded yet (a typical scenario), and does not require any aliases for dimension column names, or any data type specifications for columns.

Use the EXPORT SCHEMA command to create base tables in the schema, and export all dimension and variable information with time going down the column in fact tables. For example:

..... **Export base model to create HOLAP schema tables in RDBMS.**

schema

..... **Use SPY TERMINAL to trace to the screen the SQL commands**

..... **issued by Application Server**

spy terminal

connect Holap

prefix ABC

export schema TIME DOWN monthly YYM NAME SalesPeriod TIMETYPE Integer NTHREADS 2

spy off

end

Tip: You can get better performance if you use TIMETYPE Integer because the time field doesn't have to be converted to integer to get the ranges of time.

Exporting with time down and including aliases and datatypes

This topic explains how to export a dimensional model so that time is stored down the column in fact tables. This example assumes that the model has just been created and has no data loaded yet (a typical scenario). This example shows aliases for dimension column names and data type specifications for columns.

Procedure

1. Enter the EXPORT BASE command to create the relational tables in the schema that do not depend on specific dimension, variable, or data information. Use the TIME DOWN keyword to identify that you want to store time going down the fact table column. You must issue the EXPORT BASE command before you issue the EXPORT DIMENSIONS or EXPORT VARIABLES commands.

For example:

schema

..... **Use SPY TERMINAL to trace to the screen the SQL commands**

..... **issued by Application Server**

spy terminal

connect Holap

prefix ABC

..... **Export base model to create HOLAP schema tables in RDBMS with time down.**

EXPORT BASE TIME DOWN monthly YYM NAME SalesPeriod TIMETYPE Integer NTHREADS 2

2. Enter the EXPORT DIMENSION command to export dimensions from an Application Server dimensional model to the schema. Specify any column aliases and datatypes necessary. The

EXPORT DIMENSION command creates the Dimension Levels table and dimension tables relational tables in the schema for each dimension specified.

..... Export Product as ProductName and use short names for members

EXPORT DIMENSION Product ALIAS ProductName USENAMES

..... Export Region as RegionCode with a small integer datatype

EXPORT DIMENSION Region ALIAS RegionCode CODETYPE Smallint

..... Export Channel, Type, Account with no aliases with a small integer datatype

EXPORT DIMENSION Channel, Type, Account, CODETYPE Smallint

3. Enter the EXPORT VARIABLE command to export variables from an Application Server dimensional model to the schema. For example, to export all variables, enter the following command.

..... Export sales and costs as FLOAT

EXPORT VARIABLE Sales, Cost DATATYPE Float

..... Export margin as NUMBER

EXPORT VARIABLE Margin DATATYPE Number(10,2)

..... Export units as REAL

EXPORT VARIABLE Units DATATYPE Real

spy off

end

7.4.2.2.2 If a model already contains data

Use EXPORT DATA when you have an existing Application Server model with data in it and you have already exported the model to relational schema tables using EXPORT BASE, EXPORT DIMENSIONS, and EXPORT VARIABLES.

Use the EXPORT DATA command to export the data from an Application Server dimensional model to the schema.

For example, to export the data previously selected with Application Server SELECT statements, enter the following command:

EXPORT DATA SELECTED

Use EXPORT SCHEMA DATA when you have an existing Application Server model with data in it but you have not yet exported the model to relational schema tables. The EXPORT SCHEMA command does not let you specify column aliases or column datatypes, which are allowed if you use the EXPORT BASE command and then follow it with EXPORT DIMENSION and EXPORT VARIABLE command.

Use the EXPORT SCHEMA command with the DATA keyword to create base tables in the schema, export all dimension and variable information, and export data from an Application Server multidimensional model. For example:

EXPORT SCHEMA DATA

Note: In Hybrid OLAP models, dimension names should be not be longer than 17 characters.

7.4.2.3 Step 3 - Transforming the Source to Populate All Levels

If you built your initial model using one or more template dimensions (large that are only partially built in Application Server), you need to update those dimensions in the dimension tables to

accurately define all the members and levels of those dimensions. You use the Transformer to map to a data source and feed information into the Hybrid OLAP schema tables.

Procedure

1. Examine the dimension tables and dimension level tables created by the EXPORT command:

In Application Server, from the Window menu, choose SQL Command.

Type:

```
SELECT * FROM prefix_LEVELS_dimension  
SELECT * FROM prefix_dimension
```

2. Review the source files that contain the additional members for the dimensions.
3. Using a text editor, create a Transformer .INI file that contains a dimension section with parameters to load the additional members into the dimensions.

For example, this file will be used to populate a dimension table for the Account dimension in the RDBMS.

[Windows]

TBDB=C:\program files\Pilot Software\Common\DATA\TBDB.ENG

; This is the Transformer initialization file for Hybrid OLAP. It is used to

; set up the parameters to load dimensions and data into Hybrid OLAP models.

; The Transformer can be set up to run in batch.

; Load customers into Account dimension - assumes Link/SQL based source

[BldAccount]

Loading=Dimension

LinkIDSource=SlsBase

SourceSQL=select CUSTOMER, GROUP, CUSTLBL, GRPLBL from ACCOUNT.TXT

LinkIDTarget=SlsHOLAP

Table=Account

Schema=SLS

Truncate=Yes

Trace=Yes

Result=Yes

Format=Delimited TAB

BlockSize=8192

Blocks=100000

Buffers=10000

Level1=CUSTOMER

Level2=GROUP

Label1=CUSTLBL

Label2=GRPLBL

4. At a command prompt, run the Transformer with the initialization file and add the additional members to the Hybrid tables. Type the following command where *filename* is the name of the

Creating a Hybrid OLAP Model

Transformer initialization file and *inputmembersection* is the section name within the file containing the parameters to read in the members of the Hybrid dimensions:

SGTRANS -inifile filename.ini -s inputmembersection

5. Exit the command prompt.
6. Examine the relational tables to review the added dimension members to the updated dimension.

In Application Server, from the Window menu, choose SQL Command.

Type:

SELECT * FROM *prefix_LEVELS_dimension*

The Transformer load has updated the count of members at each level.

SELECT COUNT(*) FROM *prefix_dimension*

The resulting number of rows should be equal to the total number of members that exist in the dimension.

7.4.2.4 Step 4 - Update the Application Server Model

The original model you created is no longer valid because the template dimensions in the model do not match the dimensions in the tables in the RDBMS. Now that you have updated all the dimension member information in the dimension tables, you need to update the dimensional model to include this new information.

Procedure

1. To match the dimension information, you need to remove the original model and recreate it based on the Hybrid OLAP schema tables in the RDBMS.

.... Import model from updated schema tables in RDBMS

supervisor remove database salesdb

supervisor create database salesdb blocks 200000 members 100000 observations 1000

use salesdb

2. You import the calendar and the standard dimensions and variables from the RDBMS tables to update the Application Server model.

schema

connect HOLAP

prefix sls

import fiscal

import dimension channel

import dimension product

import dimension region

import dimension type

3. Create temporary view tables to import the output and result levels of the larger dimensions into Application Server, and then import those dimensions.

..... create views for importing output levels of Account

SQL create view tmppfx_master as select * from sls_master

SQL create view tmppfx_dimensions as select * from sls_dimensions

```

SQL create view tmppfx_levels_account as select * from sls_levels_account
SQL create view tmppfx_account as select * from sls_account where m_level > 1
prefix tmppfx
import dim account

```

- Now you import the variables and clean up the temporary tables.

```

prefix sls
import var *
..... clean up temporary tables
SQL drop view tmppfx_master
SQL drop view tmppfx_dimensions
SQL drop view tmppfx_levels_account
SQL drop view tmppfx_account
..... update columns in dimensions table used to optimize select performance –
SQL update sls_dimensions set no_guests='0' where d_name='ACCOUNT'
SQL update sls_dimensions set max_guests_level=1 where d_name='ACCOUNT'
End

```

7.4.2.5 Step 5 - Build the Tracker Table

The next step in creating a Hybrid OLAP model is to build the tracker table. The tracker table tracks the location of data in RDBMS or in Application Server and tracks how it is to be consolidated. The tracker table represents all variables and level combinations.

The tracker table, called prefix_TRACKER, is created when a model containing data is exported or when you issue the Schema command TRACKER INSERT.

Procedure

- Issue the command TRACKER INSERT * to insert records for all level combinations for all variables. Because the Tracker table stores information about the period range in the model, you must first issue a SET PERIOD command. The SET PERIOD command should cover the full range of dates you anticipate storing in the model.

- Run a procedure similar to this:

```

..... Create Tracker table entries for all dimension level combinations for
..... all variables. Starting with data for 99 through 02.
set period Jan 1999 - Dec 2002
schema
connect HOLAP
prefix sls
tracker insert *
end

```

- View the contents of the Tracker table:

- From the Window menu, choose SQL Command and verify that the Holap LinkID is selected in the lower left-hand corner of the status bar.
- On the Command line, issue the following query to show the tracker table entries for the variable sales (V_Code=1).

```
SELECT * FROM SLS_TRACKER WHERE V_CODE = 1
```

In Step 7 you will see how the tracker table fields are updated to control the consolidation process. Initially the location for the consolidated data is set to the RDBMS. You will update this field to store some of the consolidated data in the Application Server model.

7.4.2.6 Step 6 - Transforming the Source to Populate Input Data Levels

Up to this point, the relational schema tables in the RDBMS contain dimension and variable information but do not yet contain data. You use the Transformer to load all the data for the input combinations. When you load data using the Transformer, you do not need to use the Application Server-based Access LSLink/External Read command.

Procedure

1. Prepare the fact section of the Transformer initialization file to load all the data for the input combinations.
2. Go to a command prompt.
3. From the data directory, type the following command where *filename* is the name of the Transformer initialization file and *loadsection* is the section name within the file containing the parameters to load the data:

```
sgtrans -inifile filename.ini -s loadsection
```

4. Exit from the command prompt.

Here is a sample section of a Transformer initialization file used to input data into the fact tables in the RDBMS.

```
[Windows]
```

```
TBDB=C:\Pilot\DATA\TBDB.ENG
```

```
; Load the data for 99 - 02
```

```
[LoadData]
```

```
Loading=Fact
```

```
LinkIDSource=SlsBase
```

```
SourceSQL=select * from DETAIL.TXT
```

```
LinkIDTarget=HOLAP
```

```
Schema=SLS
```

```
Truncate=Yes
```

```
Trace=No
```

```
SlowLoad=No
```

```
DateFormat=YYYY/MM
```

```
Format=Delimited TAB
```

```
BlockSize=8192
```

```
Blocks=100000
```

```
Buffers=10000
```

```
VARIABLE1=UNITS,UNITS
```

```
VARIABLE2=SALES,SALES
```

Dim1=ACCOUNT,ACCOUNT

Dim2=CHANNEL,CHANNEL

Dim3=PRODUCT,PRODUCT

Dim4=REGION,REGION

Dim5=TYPE,TYPE

TIME=TIME

5. View the fact tables that are created in the RDBMS.

- From the Application Server Window menu, choose SQL Command.
- From the Edit menu, choose Select Link ID and then select HOLAP.
- From the SQL window enter:

SELECT TABLE_NAME FROM USER_TABLES

You will see the new fact tables.

7.4.2.7 Step 7 - Consolidate the Variables

Consolidating data is a two-step process. You first identify which data you want consolidated and where you want the consolidations to be stored. Then you consolidate the data.

Every level combination has a consolidation type that describes where consolidated data will be stored for that combination. When you initially load data into fact tables, all input level combinations are stored as base data in the RDBMS (consolidation type 0) to identify that the data is not consolidated.

All output level combinations have a consolidation type of preconsolidated in the RDBMS (consolidation type 1). The output level combinations are not yet consolidated though. You must mark the combinations as pending consolidation. Then when you use the CONSOLIDATE PENDING command, the specified variable data marked as pending in the tracker table will be consolidated.

You can choose whether to store the preconsolidated data in the RDBMS (consolidation type 1, the default), store the preconsolidated data in Application Server (consolidation type 6), or you may want to mark these combinations to be consolidated on the fly (consolidation type 3).

When you execute the CONSOLIDATE PENDING command, Application Server checks all rows in the Tracker table to see if data for the specified variables is marked for consolidation. If the value of the pending column in the Tracker table is set to any non-zero number, the data will be consolidated. Following successful consolidation, the value of the pending column is reset to 0, unless the NORESET keyword is specified.

This table shows the numbers associated with the various consolidation types:

CONS_TYPE	CONS_DESC
0	Base data in RDBMS
1	Preconsolidated in RDBMS
2	(Reserved for future use)
3	Consolidated on the fly
4	Attribute data in RDBMS
5	Input data at output levels in RDBMS
6	Preconsolidated data in Application Server

7.4.2.7.1 Marking the data as pending consolidation

You mark certain combinations as pending consolidation by issuing SQL code. The SQL code sets the pending flag in the Tracker table for all aggregate level combinations and indicates where the data is stored.

In the sample SQL code shown below, any non-base-data level combination that contains a level of a large dimension that is stored only in the RDBMS must also be stored in the RDBMS (cons_type = 1). All others can be stored in Application Server (cons_type = 6).

Procedure

1. Mark all combinations that are not base data as pending consolidation.

..... Consolidate sales and units variables after updating Tracker table.

schema

connect HOLAP

prefix sls

..... Pending flag set for all level combinations except base data

sql update sls_tracker set pending = 1 where cons_type <> 0

2. Set all the level combinations to a consolidation type of six so that all output levels are stored preconsolidated in Application Server.

..... Store all level combinations in Application Server (cons_type of 6)

sql update sls_tracker set cons_type = 6 where cons_type <> 0

3. For the combinations containing data for the levels of the dimensions known only in the RDBMS, change the consolidation type to 1 so that data will be stored preconsolidated in the RDBMS.

..... Store all level combinations with input of big dimension in relational (cons_type of 1)

sql update sls_tracker set cons_type = 1 where l_account = 1

end

Now you are ready to consolidate the data marked as pending.

7.4.2.7.2 Consolidating the data

Use the CONSOLIDATE PENDING command to consolidate any variable data marked as pending in the tracker table. When you execute the CONSOLIDATE PENDING command, Application Server checks all rows in the Tracker table to see if data for the specified variables is marked for consolidation. If the value of the pending column in the Tracker table is set to any non-zero number, the data will be consolidated. Following successful consolidation, the value of the pending column is reset to 0, unless the NORESET keyword is specified.

Procedure

1. Create a procedure similar to this one to consolidate the variables in the model. You can include the SPY and PROGRESS commands to examine the SQL code generated for the consolidate.

..... Consolidate sales and units variables after updating Tracker table.

schema

connect HOLAP

prefix sls

spy terminal

progress on

consolidate pending noforeignkeys unrecoverable noanalyze sales nthreads 2

consolidate pending noforeignkeys unrecoverable nodimrefresh units nthreads 2

progress off

spy off

end

2. Use the Rollup Editor to verify the location of data in Application Server. For example:

ROLLUP Sales

ADD EVERYBODY

SHOW COUNT

END

You can see that the Application Server model contains data for the output combinations only of the large dimensions. All of the combinations that contain input data for this dimension are stored in the fact tables in the RDBMS.

3. Use the SQL window to view the location of data in Hybrid OLAP and see the record count for the level combinations stored in the RDBMS.

From the Window menu, choose | SQL Command.

From the SQL window enter:

SELECT * FROM SLS_TRACKER WHERE V_CODE=1

Scroll until the NROWS column is visible. Cons_Type = 0 means base data is stored in RDBMS, Cons_Type = 1 means consolidated data is stored in the RDBMS, Cons_Type = 6 means consolidated data stored in Application Server.

7.4.2.8 Adding New Data to the RDBMS

Procedure

1. Review the source file containing the new period of data.
2. From a command prompt go to the Hybrid OLAP data directory.
3. Using a text editor, review the section in the Transformer .INI file that loads data for the new period. For example:

[Loadnewperiod]

Loading=Fact

LinkIDSource=Base

SourceSQL=select * from DETAILMay2002.TXT

LinkIDTarget=HOLAP

Schema=SLS

Truncate=No

Trace=Yes

SlowLoad=No

DateFormat=YYYY/MM

Period=2002/05

Format=Delimited TAB

Creating a Hybrid OLAP Model

BlockSize=8192

Blocks=100000

Buffers=10000

VARIABLE1=UNITS,UNITS

VARIABLE2=SALES,SALES

Dim1=ACCOUNT,ACCOUNT

Dim2=CHANNEL,CHANNEL

Dim3=PRODUCT,PRODUCT

Dim4=REGION,REGION

Dim5=TYPE,TYPE

TIME=TIME

4. Run the Transformer with the initialization file to load a new period of input or base data for the Hybrid model. Type the following command to load new data where *filename* is the Transformer initialization file and *newperiodsection* is the section name in the initialization file containing the parameters to load the data from the source file:

SGTRANS -INIFILE *filename*.INI -S *newperiodsection*

5. Use the SQL window to look at delta table:

From the Window menu, choose | SQL Command.

From the SQL window enter:

SELECT * FROM SLS_F_1_36_F_D

6. Issue the IDQL command:

CONSOLIDATE INCREMENTAL SALES, UNITS

7.5 Using Fact Tables in External Applications

The default schema fact tables are built with time going across. Optionally, Hybrid OLAP's open architecture allows you to store time-related information down the column in fact tables. This format is common in other applications, and allows you to use the fact tables in those applications if desired.

To create a model that can be used in other applications, you follow the same steps as though you were building a standard Hybrid OLAP model. When you are exporting the model, you include the TIME DOWN keywords in the command to specify that you want to store the time-related information down the columns in the fact tables.

You can export a dimensional model with time going down columns in two ways:

Export without specifying any aliases for dimension column names, or any data type specifications for columns. Use the EXPORT SCHEMA command with the TIME DOWN keyword for this method.

Export while specifying aliases for dimension column names and data type specifications for columns. Use the EXPORT BASE, EXPORT DIMENSIONS, and EXPORT VARIABLES commands with the TIME DOWN keyword on the EXPORT BASE command.

You include the TIME DOWN keyword on only one EXPORT command to allow the fact table configuration. All proceeding EXPORT commands assume the time down configuration.

7.6 Working with SCHEMA Subsystem

Using Hybrid OLAP from the IDQL command line

The SCHEMA subsystem in Application Server consists of a series of commands that allow Application Server users to build and maintain schema models from the IDQL command line.

A Hybrid OLAP schema is created when the components of an Application Server model (such as dimensions, variables, and data) are exported to relational tables.

As in other Application Server subsystems, you can use many generic Application Server commands while you are working in the SCHEMA subsystem. Some Application Server commands behave differently in Hybrid OLAP models.

After you have enabled Hybrid OLAP, you can enter the SCHEMA subsystem. The SCHEMA subsystem consists of a series of commands that allow Application Server users to build and maintain schema models from the IDQL command line.

Entering the SCHEMA subsystem

Enter the SCHEMA command to enter the SCHEMA subsystem. For example:

```
SCHEMA
```

Note: Before entering the SCHEMA subsystem, make sure you set the database to exclusive use. For example, to set a database called MARKETING to exclusive use, enter the following command:

```
USE MARKETING EXCLUSIVE
```

Connecting to a Link ID data source

Enter the CONNECT command to specify the Link ID you are using to connect to the relational database. For example, to connect to a relational data source using a Link ID called "dblink1", enter the following command:

```
CONNECT dblink1
```

Importing

Every relational schema that you create must have a prefix. The schema prefix must be of 5 byte or fewer, and begin with an alphabetical character. If you have long dimension names (that is, dimension names of 10 or more characters, up to a maximum of 17), it is advisable to use a prefix shorter than 10 characters. The schema prefix is applied to all of the relational tables in a schema, and serves as a unique identifier.

Setting the schema prefix

Enter the PREFIX command to set the schema prefix. For example, to set the prefix "retail", enter the following command:

```
PREFIX retail
```

Viewing the SQL generated by Application Server

Enter the SPY command to view the SQL generated by Application Server. For example, to send the SQL to the output window, enter the following command:

```
SPY TERMINAL
```

To stop capturing the SQL output, enter the following command:

```
SPY OFF
```

Issuing SQL while you are in the SCHEMA subsystem

You can issue SQL statements to be passed directly to the RDBMS while you are in the SCHEMA subsystem.

Enter the SQL command to direct SQL statements to the RDBMS. For example:

```
SQL UPDATE TABLE DEMO_REFERENCE SET A=1 WHERE B=2
```

Where required by your RDBMS, you must follow the SQL command with a commit. For example:

```
SQL COMMIT
```

Note: You cannot enter SQL SELECT statements using the SQL (Schema) command.

Exiting

Enter the END command to leave the SCHEMA subsystem.

```
END
```

7.7 Importing

Use the IMPORT FISCAL command to import fiscal calendar information defined in a schema into an existing Application Server dimensional model. For example:

```
IMPORT FISCAL
```

Note: When a fiscal calendar is already set in the Application Server dimensional model, and variable data exists, the existing fiscal calendar is used.

Enter the IMPORT DIMENSION command to import dimensions defined in a schema into an existing Application Server dimensional model. For example, to import all of the schema dimensions into the model, enter the following command:

```
IMPORT DIMENSION *
```

Enter the IMPORT VARIABLES command to import variables defined in a schema into an existing Application Server dimensional model. For example:

```
IMPORT VARIABLES
```

Enter the IMPORT SCHEMA command to import fiscal calendar information, dimensions, and variables defined in a schema into an existing Application Server dimensional model. For example:

IMPORT SCHEMA

7.8 Deleting a Schema

Procedure to delete a schema in a single step

Enter the DROP SCHEMA command to delete all of the schema tables in a single step:

DROP SCHEMA

During the design and prototyping stage of data mart development, you may choose to delete part of a schema rather than the whole thing. Tables must be removed from the schema in the order designated below.

This command removes the Reference table(s) from the schema:

DROP REFERENCES

This command removes the Tracker table and Fact table(s) from the schema:

DROP DATA

This command removes all of the Dimension tables from the schema:

DROP DIMENSION *

This command removes all of the variables from the schema

DROP VARIABLE *

This command removes the base tables from the schema:

DROP BASE

7.9 Viewing Definitions

Enter the VIEW DIMENSION command to view the schema dimensions. For example:

VIEW DIMENSION

Enter the VIEW VARIABLE command to view the schema variables. For example:

VIEW VARIABLE

Enter the VIEW SCHEMA command to view information about the variables and dimensions defined in the current schema. For example:

VIEW SCHEMA

Enter the VIEW PREFIX command to view the current schema prefix. For example:

VIEW PREFIX

Enter the VIEW CACHE command to view the current cache settings. For example:

VIEW CACHE

Enter the VIEW CONNECTION command to view the Link ID being used to connect to the relational database. For example:

VIEW CONNECTION

7.10 Working with Dimensions, Variables and Data

You can define multiple labels for a dimension member in a relational schema.

Enter the SET LABEL command to define more than one label for a dimension member. For example, to use the label set alt3 for the Scenario dimension, while connecting to the relational database with the Link ID dblink1, enter the following command:

SET LABEL Scenario FROM dblink1 NAME alt3

The Drillthru attribute specifies that a variable can be stored in either the relational tables or the Application Server multidimensional model.

Enter the SET VARIABLE command to apply the Drillthru attribute to a variable. For example, to set all of the variables in a model as Drillthru, enter the following command:

SET VARIABLE *

Enter the SET VARIABLE command with the NOFROM keyword. For example, to remove the Drillthru attribute to all variables in a model, enter the following command:

SET VARIABLE * NOFROM

Enter the SHOW VARIABLE command to view the variables. For example:

SHOW VARIABLE

Drillthru variables are marked with a "D" in the Type column. In addition, after each Drillthru variable, the Link ID and schema prefix are listed.

Enter the CACHE command to control how often Application Server goes back to the RDBMS to refresh the control information that it needs. For example, to cache Tracker table data until the next Application Server command is issued, enter the following command:

CACHE TRACKER COMMAND

8 Schema Subsystem Command Reference

8.1 CACHE (Schema)

Use

CACHE is a Hybrid OLAP SCHEMA subsystem command. CACHE controls how often Application Server goes back to the RDBMS to refresh the control information that it needs. Certain combinations are not supported.

Syntax

CACHE {[ALL | DATA | DIMENSIONS | TRACKER | VARIABLES] [DELETE | FOREVER | SESSION | COMMAND | MATRIX | TIME <seconds> | OFF]}

Parameter	Description
ALL	Sets the caching options for data, dimensions, Tracker table, and variable information read into Application Server from the schema.
DATA	Sets the caching options for data read into Application Server from the schema.
DIMENSIONS	Sets the caching options for dimension information read into Application Server from the schema.
TRACKER	Sets the caching options for Tracker table information read into Application Server from the Tracker table in the schema.
VARIABLES	Sets the caching options for variable information read into Application Server from the schema.
DELETE	Deletes the cached data for the specified cache type.
FOREVER	Retains the cached data for the specified cache type until it is deleted or the EXIT CLEAR command is executed.
SESSION	Retains the cached data for the specified cache type until it is deleted or the Application Server session is terminated.
COMMAND	Retains the cached data for the specified cache type until the next Application Server command is executed.
MATRIX	Caches the data for the specified cache type on a matrix basis.
TIME <seconds>	Specifies in seconds how often the SAP NetWeaver BI Connector checks to see if new data has been loaded. The default is 600 seconds (10 minutes).
OFF	Switches off caching for the specified cache type.

Remarks

You use CACHE command keywords in pairs. The first keyword specifies the type of data to be cached. The second keyword determines how the data is to be cached. The following table shows the supported and default caching option for each data type. You may need to maximize the Help window to view all of the columns in the table.

Data type	Supported values	Default value
DATA	COMMAND	COMMAND
DIMENSIONS	COMMAND, SESSION, FOREVER	FOREVER
TRACKER	COMMAND	COMMAND
VARIABLES	COMMAND, SESSION, FOREVER	COMMAND
TABLES	COMMAND, SESSION, FOREVER	SESSION

8.2 CONSOLIDATE {PENDING | INCREMENTAL} (Schema)

Use

CONSOLIDATE {PENDING | INCREMENTAL} is a Hybrid OLAP SCHEMA subsystem command. CONSOLIDATE PENDING consolidates specified variable data that is marked as pending in the Tracker table. CONSOLIDATE INCREMENTAL allows you to make an intelligent consolidation of only those variable/level combinations that require consolidation following a change in input data values or dimension structure.

Syntax

```
CONSOLIDATE {PENDING | INCREMENTAL}
  [ARRAYSIZE <number>]
  [DIRECTLOAD]
  [DOINSERTS FIRST | DOUPDATES FIRST]
  [INDEXSPACE <tablespace_name>]
  [NOANALYZE]
  [NOCOMPRESS]
  [NODIMREFRESH <dimension>]
  [NOFOREIGNKEYS]
  [NORESET [<variables>] ]
  [NTHREADS <number>]
  [TABLESPACE <tablespace_name>]
  [UNRECOVERABLE]
  [WORKDB <work_database>]
```

Parameter	Description
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.
DIRECTLOADPATH	(Oracle) Invokes the Oracle Sql*Loader and uses the DIRECT load path to load fact tables, create the primary key, and populate any other indexes at table creation time.
DOINSERTS FIRST	Specifies that you want Application Server to try to insert rows first during a CONSOLIDATE PENDING or CONSOLIDATE INCREMENTAL. If you know that your data will require mostly inserts (i.e., you have all new data for new time periods, and there are few changes to existing data), you should perform a DOINSERTS to improve performance. Tip: If you have time stored down the fact tables, you should perform a DOINSERTS because it is more likely that data will be inserted more than updated for new time periods. By default, if you do not specify either DOINSERTS or DOUPDATES, the CONSOLIDATE PENDING command will try inserts first when the Schema is storing time down, and it will try updates first when the Schema is storing time across. If the CONSOLIDATE PENDING determines that the target RDBMS table is empty then it will always do inserts.
DOUPDATES FIRST	Specifies that you want Application Server to try to update rows first during a CONSOLIDATE PENDING or INCREMENTAL, and then insert rows as necessary. If you know that your data will require mostly updates (i.e., you are consolidating data that has changed for the same time periods, but you don't have data for new time periods), you should perform a DOUPDATES FIRST to help improve performance. Tip: When you have time stored across the fact tables, Application Server needs to do updates and inserts to fully account for updating values in the rows before inserting new data. If you know that your data will require mostly updates, such as

when time is going across, you should perform a DOUPDATES to help improve performance.

By default, if you do not specify either DOINSERTS or DOUPDATES, the CONSOLIDATE PENDING command will try inserts first when the Schema is storing time down, and it will try updates first when the Schema is storing time across. If the CONSOLIDATE PENDING determines that the target RDBMS table is empty then it will always do inserts.

INDEXSPACE <tablespace_name>

Creates the indexes on this tablespace (Oracle or DB2). INDEXSPACE <tablespace_name> allows you to control the placement of indexes.

NOANALYZE

(Oracle) Do not run the ANALYZE command to gather statistics for the cost-based optimizer.

NODIMREFRESH <dimension>

Information for the specified dimension is not read from the relational schema. The existing dimension cache will be used instead, improving performance significantly.

Notes:

Use NODIMREFRESH only when you are sure that the specified dimension has not changed since the most recent CONSOLIDATE PENDING.

NODIMREFRESH information is stored in the Work database. If you use a custom Work database specified with the WORKDB keyword, you will be able to use the NODIMREFRESH information across sessions. If you use the default Work database instead and want to reuse NODIMREFRESH, make sure you save your Work database when you exit Application Server.

NOCOMPRESS

Turns off the default compression used by Hybrid OLAP on cached consolidated data of types 1, 2, and 6 before it is returned to the RDBMS.

Note: The NOCOMPRESS keyword is intended for experimental use. In general, the default compression is recommended.

NOFOREIGNKEYS

Foreign key integrity constraints are not created.

NORESET

Do not reset the pending column in the Tracker table after data is consolidated. This option can be useful during testing.

<variables>

One or more variable names separated by commas. If you do not specify a variable name (the default), the currently selected variables are consolidated. Specify variable names that use special characters in single quotation marks (' ').

NTHREADS <number>

Allows you to run two processes at the same time. When <number> is greater than 1, this keyword allows one thread to carry out the Application Server and sgtrans processing and another thread to carry out the RDBMS processing concurrently.

For example, while a CONSOLIDATE PENDING command is executing, Application Server can read rows from the RDBMS and aggregate those rows at the same time that the RDBMS is delivering the rows.

Tip: On multiprocessor systems, using multiple threads concurrently should always improve throughput during reading and writing processes. On single processor systems, it may also improve throughput if a chunk of any elapsed time is I/O during which the CPU would otherwise be idle.

If you omit the NTHREADS keyword, or if you specify NTHREADS 1, only one process will occur at a time. For example, while a CONSOLIDATE PENDING command is executing, Application Server will read the rows from the RDBMS (via Link). While the RDBMS delivers the rows, Application Server remains idle waiting until the RDBMS is finished. When the rows are delivered, Application Server starts aggregating, while the RDBMS remains idle waiting for the next request.

TABLESPACE <tablespace_name>

DELTA (Schema)

	Puts any created Fact tables on this tablespace (Oracle or DB2). TABLESPACE <tablespace_name> allows you to control the placement of data.
UNRECOVERABLE	(Oracle) Create all indexes without line redo log writes.
WORKDB <work_database>	Sets an alternate Application Server database as the Work database. This is particularly useful in large models, when the Application Server Work database might be too small.
	Note: When specified, WORKDB must be the first keyword after PENDING.

Remarks

When you execute the CONSOLIDATE PENDING command, Application Server checks all rows in the Tracker table to see if data for the specified variables is marked for consolidation. If the value of the *pending* column in the Tracker table is set to any non-zero number, the data will be consolidated. Following successful consolidation, the value of the *pending* column is reset to 0, unless the NORESET keyword is specified.

Data will be marked as pending consolidation when the consolidation type of previously unconsolidated data is changed to a consolidation type that requires it to be stored in preconsolidated format (in either Application Server or the relational database).

8.3 DELTA (Schema)

Use

DELTA is a Hybrid OLAP SCHEMA subsystem command that allows you to delete, drop, or replace all of the delta rows from your schema's Dimension Delta or Fact Delta table(s).

Syntax

DELTA {TRUNCATE | DROP | REPLACE} {DIMENSION | FACT | BOTH} { * | <list of table names> }

Parameter	Description
TRUNCATE	Truncates the delta rows from the specified Delta tables.
DROP	Drops the delta rows from the specified Delta tables.
REPLACE	Replaces the delta rows in the specified Delta tables.
DIMENSION	Truncate, drop, or replace only the Dimension Delta table delta rows.
FACT	Truncate, drop, or replace only the Fact Delta table delta rows.
BOTH	Truncate, drop, or replace both Dimension Delta and Fact Delta table delta rows.
*	Indicates that the DELTA command applies to all Delta tables of the specified type.
<list of table names>	For Dimension Delta tables, specify a list of table names. For Fact Delta tables, specify a list of table names, including the prefix if necessary.

Remarks

You should truncate delta rows from Fact Delta tables if you have done a BCP or a Sql*Loader Direct load, a TRUNCATE table, or when a CONSOLIDATE INCREMENTAL fails.

8.4 DROP BASE (Schema)

Use

DROP BASE is a Hybrid OLAP SCHEMA subsystem command. DROP BASE deletes the base tables from the current schema

Syntax

DROP BASE

Remarks

You should run the DROP REFERENCES, DROP DATA, DROP DIMENSION, and DROP VARIABLE commands before you use the DROP BASE command to delete base tables from the schema. Alternatively, you can run the DROP SCHEMA command to delete the entire schema in a single step.

The DROP BASE command deletes the following relational tables from the schema in the order shown:

Periodicities table

Consolidations table

Dimension Types table

Dimensions table

By Dimensions table

Format table

Variables table

Master table

8.5 DROP DIMENSION (Schema)

Use

DROP DIMENSION is a Hybrid OLAP SCHEMA subsystem command. DROP DIMENSION deletes tables for the specified dimensions from the current schema.

Syntax

DROP DIMENSION <dimensions> [ALL | REGULAR | STRUCTURED | UNSTRUCTURED | ATTRIBUTE]

Parameter	Description
<dimensions>	One or more dimension names separated by commas. You can specify an asterisk character (*) to delete tables for all dimensions defined in the schema.
ALL	Deletes tables for all dimensions in the schema.
REGULAR	Deletes tables for dimensions that are not attribute or hierarchical attribute dimensions.
STRUCTURED	Deletes tables for hierarchical attribute dimensions.
UNSTRUCTURED	Deletes tables for non-hierarchical attribute dimensions.
ATTRIBUTE	Deletes tables for all attribute dimensions, regardless of whether they are hierarchical.

Remarks

You should run the DROP REFERENCES and DROP DATA commands before you use the DROP DIMENSION command to delete tables for dimensions in the schema. Alternatively, you can run the DROP SCHEMA command to delete the entire schema in a single step.

8.6 DROP REFERENCES (Schema)

Use

DROP REFERENCES is a Hybrid OLAP SCHEMA subsystem command. DROP REFERENCES drops any Reference tables from the existing schema.

DROP SCHEMA (Schema)

Syntax

DROP REFERENCES

Remarks

You should use the DROP REFERENCES command to delete Reference tables from the schema, before selectively deleting other tables from the schema. Alternatively, you can run the DROP SCHEMA command to delete the entire schema in a single step.

8.7 DROP SCHEMA (Schema)

Use

DROP SCHEMA is a Hybrid OLAP SCHEMA subsystem command. DROP SCHEMA deletes the current schema.

Syntax

DROP SCHEMA

Remarks

DROP SCHEMA deletes all the tables in the current schema without requesting user confirmation.

8.8 DROP VARIABLE (Schema)

Use

DROP VARIABLE is a Hybrid OLAP SCHEMA subsystem command. DROP VARIABLE deletes the specified variables from the current schema.

Syntax

DROP VARIABLE <variables>

Parameter	Description
<variables>	One or more variable names separated by commas. You can specify an asterisk (*) character to delete all variables defined in the schema.

Remarks

You should run the DROP REFERENCES, DROP DATA, and DROP DIMENSION commands before you issue the DROP VARIABLE command to drop variables from the schema. Alternatively, you can run the DROP SCHEMA command to delete the entire schema in a single step.

8.9 EXPORT BASE (Schema)

Use

EXPORT BASE is a Hybrid OLAP SCHEMA subsystem command. EXPORT BASE creates the relational tables in the schema that do not depend on specific dimension, variable, or data information.

You can export the schema with time going across, or time going down. If you export the schema with time going down the columns in the fact tables, you have the flexibility to use the tables in other applications that also support this common format for their fact tables. **Tip:** Store time going down the table if you want to optimize the time it takes to load and consolidate data. Store time going across the table if you want to optimize end-user performance.

Syntax

EXPORT BASE

```
[ARRAYSIZE <number>]
[INDEXSPACE <tablespace_name>]
[NOANALYZE]
[NOFOREIGNKEYS]
[TABLESPACE <tablespace_name>]
{ TIME ACROSS [ V_CODE <text> ] [USENAMES ] |
  TIME DOWN <periodicity> <dateformat> NAME <colname> TIMETYPE <type> } }
[UNRECOVERABLE]
```

Parameter	Description																		
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.																		
INDEXSPACE <tablespace_name>	Creates the indexes on the specified tablespace (Oracle or DB2). This allows you to distribute indexes across many disks.																		
NOANALYZE	For Oracle, the NOANALYZE keyword omits running the ANALYZE command to gather statistics for the cost-based optimizer. For DB2, the NOANALYZE keyword omits running the RUNSTATS command to gather statistics for the cost based optimizer. Note: Use this option with care. It is important that the statistics on tables be kept up to date, since this is the basis on which the RDBMS decides how to execute queries. If statistics are not available, or out of date, this may lead to the RDBMS choosing poor execution plans for queries leading to poor performance.																		
NOFOREIGNKEYS	Foreign key integrity constraints are not created.																		
TABLESPACE <tablespace_name>	Creates the base tables on the specified tablespace (Oracle or DB2). This allows you to distribute tables across many disks.																		
TIME ACROSS	Specifies that the schema will be exported with time going across the fact tables. This is the default setting.																		
V_CODE <text>	Uses the value of <text> as the column name for variables in fact tables instead of the word V_CODE when time is stored across the fact table.																		
USENAMES	Uses short names instead of codes in fact tables.																		
TIME DOWN	Specifies that the schema will be exported with time going down the fact tables. Specify a repeating clause of <periodicity> <dateformat> NAME <colname> TIMETYPE <datatype>, one for each different periodicity that is going to be present in fact tables. When you specify TIME DOWN, a new row is entered in the prefix_MASTER file that looks like this: <table> <tr> <td>MST_KEY</td><td>MST_SETTING</td></tr> <tr> <td>TIME_ORIENTATION</td><td>DOWN</td></tr> </table>	MST_KEY	MST_SETTING	TIME_ORIENTATION	DOWN														
MST_KEY	MST_SETTING																		
TIME_ORIENTATION	DOWN																		
<periodicity>	One of the following: <table> <tr> <td>Yearly</td><td>12 months</td></tr> <tr> <td>Semiannual</td><td>6 months</td></tr> <tr> <td>Quarterly</td><td>quarter</td></tr> <tr> <td>Bimonthly</td><td>2 months</td></tr> <tr> <td>Monthly</td><td>month</td></tr> <tr> <td>Lunar</td><td>28 days. Use this with a 13-month fiscal year only.</td></tr> <tr> <td>Weekly</td><td>week</td></tr> <tr> <td>Biweekly</td><td>2 weeks</td></tr> <tr> <td>Daily</td><td>day</td></tr> </table>	Yearly	12 months	Semiannual	6 months	Quarterly	quarter	Bimonthly	2 months	Monthly	month	Lunar	28 days. Use this with a 13-month fiscal year only.	Weekly	week	Biweekly	2 weeks	Daily	day
Yearly	12 months																		
Semiannual	6 months																		
Quarterly	quarter																		
Bimonthly	2 months																		
Monthly	month																		
Lunar	28 days. Use this with a 13-month fiscal year only.																		
Weekly	week																		
Biweekly	2 weeks																		
Daily	day																		
<dateformat>	Type of data the field contains: <table> <tr> <td><text></td><td>Specifies that the TIME column contains all characters. Supplying a text string of XXX for example, means that the TIME column will be</td></tr> </table>	<text>	Specifies that the TIME column contains all characters. Supplying a text string of XXX for example, means that the TIME column will be																
<text>	Specifies that the TIME column contains all characters. Supplying a text string of XXX for example, means that the TIME column will be																		

EXPORT BASE (Schema)

		character columns with values XXX1, XXX2, XXX3. The number is the period number (relative to the earliest period).
ORDINAL		Specifies that you just want the relative period number without any text prefix in the TIME column. This way, the TIME column contains just the period number, for example 1, 2, 3.
<date>		One of these: myy , ymd , yymd , dmyy , dmy , mdyy , mdy , ydm , yydm , yym , my , or ym . Specifies that the columns will be in date format. If the year is four digits (2002), use a format with yy . If the year is two digits (02), use a format with y . If the month is alphabetic (May 2000), replace the letter m with the word month , for example monthyy . Or specify Month/d/y for a date of Dec/3/02. Note: Oracle supports character translation with month names in its TO_DATE function so it is acceptable to use month names if using Oracle. If you are using an RDBMS other than Oracle, you should avoid using month names if possible. No other RDBMS supports character translation; using month names will cause the SQL to become complex and most likely perform poorly. You can use the separator characters dash (-), slash(/), or period (.) in the date format where appropriate. For example, specify m-d-y for a date of 12-3-02 and specify m/d/y for a date of 12/3/02.
NAME <colname>		Specify the time column name for that periodicity.
TIMETYPE <type>		Specify the RDBMS datatype of the time column.
SMALLINT	2 bytes	A short integer between -32,768 and 32,767.
INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.
NUMBER(p,s)		Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.
CHAR(<n>)		A fixed length character string using <n> bytes.
VARCHAR(<n>)		A varying length character string using up to <n> bytes.
DATE		A date string stored in the internal date format of the RDBMS.
DATETIME		A date and time string stored in the internal date format of the RDBMS.
Notes:		
Oracle stores any numeric datatypes in packed decimal varying length data internally so SMALLINT, INTEGER, NUMBER(p,s), REAL, and FLOAT are all the same. If using DB2, all the datatypes apply.		
UNRECOVERABLE		(Oracle) Create all indexes without line redo log writes.

Remarks

When you create a schema manually from the IDQL command line in Application Server, you must run the EXPORT BASE command before you can export dimensions, variables, or data. The EXPORT BASE command creates the following relational tables in the schema in the order shown:

Master table

Periodicities table

Consolidations table

Dimension Types table

Dimensions table

By Dimensions table

Format table

Variables table

Notes:

The time orientation is set by the first EXPORT command used. If you don't specify any keywords, then TIME ACROSS is used. If you use a TIME ACROSS or DOWN keyword on any other EXPORT command, the time set by the first command will be applied.

If you use the EXPORT BASE command, you must then also use the EXPORT DIMENSION and EXPORT VARIABLES commands to export those components individually.

If you already have fact tables, the datatypes and date formats should be set to match those which exist in those tables. Hybrid OLAP handles the information appropriately. If you are creating fact tables for the first time and you want to store time information going down the column, it is important to consider the appropriate datatypes and date format to use. You should choose a datatype and date format that leads to the simplest and most efficient SQL. For example choose an INTEGER datatype with a format YYMD or DATE or DATETIME. These datatypes result in simple SQL, which the RDBMS handles well. If you have date values in character columns with optional separators, when HOLAP has to query fact tables for data between two given dates, the SQL must use the RDBMS functions to translate the character strings to dates, which is extra work. It may also mean that the RDBMS cannot use an index on the fact table to its best advantage.

8.10 EXPORT DATA (Schema)

Use

EXPORT DATA is a Hybrid OLAP SCHEMA subsystem command. EXPORT DATA exports the specified data from an Application Server database to the schema.

Syntax

```
EXPORT DATA { SELECTED | <variables> }
  [ARRAYSIZE <number>]
  [COMMIT [EVERY] <rows>]
  [DATATYPE <datatype>]
  [DIRECTLOADPATH]
  [INDEXSPACE <tablespace_name>]
  [NOANALYZE]
  [NOFOREIGNKEYS]
  [NOSORT]
  [NTHREADS <number>]
  [UNRECOVERABLE]
  [TABLESPACE <tablespace_name>]
```

Parameter	Description
SELECTED	Exports the data currently selected in the Application Server database.
<variables>	One or more variable names separated by commas. You can also specify an asterisk character (*) to export data for all variables defined in an Application Server database.
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.
COMMIT	Determines that a SQL COMMIT command should be executed after a specified number of data rows have been read.
EVERY	May be added after the COMMIT keyword to improve readability.
<rows>	The number of rows after which a SQL COMMIT command should be executed when reading data. The default is 1000 rows.

EXPORT DATA (Schema)

DATATYPE <type>	(Available only when using EXPORT BASE TIME ACROSS) Use DATATYPE to specify the RDBMS datatype of the data column in cases where you exported variables in a time across orientation without specifying individual datatypes.	
TINYINT	1 byte	An integer value between 0 and 255.
SMALLINT	2 bytes	A short integer between -32,768 and 32,767.
INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.
REAL	4 bytes	A single-precision floating-point value with a range of 3.402823E38 to 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.
FLOAT	8 bytes	A double-precision floating-point value with a range of 1.79769313486232E308 to 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.
NUMBER(p,s)		Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.
DIRECTLOADPATH	Invokes the Oracle SQL*Loader and uses the DIRECT load path to load fact tables, create the primary key, and populate any other indexes at table creation time.	
INDEXSPACE <tablespace_name>	Create the indexes on this tablespace (Oracle or DB2). Used with EXPORT DATA SELECTED, INDEXSPACE <tablespace_name> allows you to distribute indexes across many disks.	
NOANALYZE	For Oracle, the NOANALYZE keyword omits running the ANALYZE command to gather statistics for the cost-based optimizer. For DB2, the NOANALYZE keyword omits running the RUNSTATS command to gather statistics for the cost based optimizer. Note: Use this option with care. It is important that the statistics on tables be kept up to date, since this is the basis on which the RDBMS decides how to execute queries. If statistics are not available, or out of date, this may lead to the RDBMS choosing poor execution plans for queries leading to poor performance.	
NOFOREIGNKEYS	Foreign key integrity constraints are not created.	
NOSORT	(Oracle) Normally Application Server will write out data to the RDBMS in the order of any primary key in the table. By presenting data presorted to the RDBMS, index creation is faster in the RDBMS because the RDBMS does not have to presort data itself. Specify NOSORT if you do not want Application Server to presort, but instead have the RDBMS sort itself when creating an index.	
NTHREADS <number>	Allows you to run two processes at the same time when <number> is greater than 1. NTHREADS 2 allows one thread to carry out the Application Server and sgtrans processing and another thread to carry out the RDBMS processing concurrently. If you omit the NTHREADS keyword, or if you specify NTHREADS 1, only one process will occur at a time. For example, during an EXPORT DIMENSION or EXPORT DATA command, Application Server is reading dimensions members or time series respectively, collecting the information into batches of rows, and then sending a batch to the RDBMS for insertion into a dimension or fact table. In a single thread, Application Server has to wait while the RDBMS does the inserts before it prepares the next batch. If NTHRAD is set to 2 and two threads	

are running, Application Server can be preparing the next batch of rows to give to the RDBMS while the RDBMS is inserting the last batch of rows.

Tip: On multiprocessor systems, using multiple threads concurrently should always improve throughput during reading and writing processes. On single processor systems, it may also improve throughput if a chunk of any elapsed time is I/O during which the CPU would otherwise be idle.

TABLESPACE <tablespace_name>

Create the tables on this tablespace (Oracle or DB2). Used with EXPORT DATA SELECTED, TABLESPACE <tablespace_name> allows you to distribute tables across many disks.

UNRECOVERABLE

(Oracle) Create all indexes without line redo log writes.

Remarks

When you create a schema manually from the IDQL command line in Application Server, you must run the EXPORT BASE, EXPORT VARIABLE, and EXPORT DIMENSION commands before you can export data. Alternatively, you can run the EXPORT SCHEMA command to execute these commands in a single step.

The EXPORT DATA command creates the following tables:

Tracker table

Fact tables

8.11 EXPORT DIMENSION (Schema)

Use

EXPORT DIMENSION is a Hybrid OLAP SCHEMA subsystem command. EXPORT DIMENSION creates the relational tables in the schema that relate to specific dimensions in the Application Server database.

You export the dimensions separately so you can give the corresponding fact table column a name via the ALIAS <name> keyword and a datatype and tell it to use the short name instead of a member code.

Syntax

```
EXPORT DIMENSION [REGULAR|STRUCTURED|UNSTRUCTURED|ATTRIBUTE] [UNRECOVERABLE]
    <dimensions>
    [ALIAS <name>]
    [ARRAYSIZE <number>]
    [DIRECTLOADPATH]
    [INDEXSPACE <tablespace_name>]
    [NOANALYZE]
    [NOFOREIGNKEYS]
    [NOSORT]
    [NTHREADS <number>]
    [TABLESPACE <tablespace_name>]
    [USENAMES | CODETYPE <datatype> ]
```

Parameter	Description
REGULAR	Exports dimensions that are not attribute or hierarchical attribute dimensions.
STRUCTURED	Exports only hierarchical attribute dimensions.
UNSTRUCTURED	Exports only non-hierarchical attribute dimensions.
ATTRIBUTE	Exports all attribute dimensions, regardless of whether they are hierarchical.
<dimensions>	One or more dimension names separated by commas. You can specify an asterisk character (*) to export all dimensions defined in an Application Server database.
UNRECOVERABLE	(Oracle) Create all indexes without line redo log writes.

EXPORT DIMENSION (Schema)

ALIAS <name>	<p>Uses the value for <name> as the fact column name rather than the dimension name.</p> <p>Tip: By adding column names that are familiar to you in the fact tables, it is much easier to navigate through the fact tables in an ad hoc manner using SQL code.</p>
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.
DIRECTLOADPATH	Invokes the Oracle SQL*Loader and uses the DIRECT load path to load fact tables, create the primary key, and populate any other indexes at table creation time. For DB2, invokes the DB2 LOAD facility.
INDEXSPACE <tablespace_name>	Create the indexes on this tablespace (Oracle or DB2). Used with EXPORT DIMENSION, INDEXSPACE <tablespace_name> allows you to distribute indexes across many disks.
NOANALYZE	For Oracle, the NOANALYZE keyword omits running the ANALYZE command to gather statistics for the cost-based optimizer. For DB2, the NOANALYZE keyword omits running the RUNSTATS command to gather statistics for the cost based optimizer. Note: Use this option with care. It is important that the statistics on tables be kept up to date, since this is the basis on which the RDBMS decides how to execute queries. If statistics are not available, or out of date, this may lead to the RDBMS choosing poor execution plans for queries leading to poor performance.
NOFOREIGNKEYS	Foreign key integrity constraints are not created.
NOSORT	<p>(Oracle) Normally Application Server will write out data to the RDBMS in the order of any primary key in the table. By presenting data presorted to the RDBMS, index creation is faster in the RDBMS because the RDBMS does not have to presort data itself.</p> <p>Specify NOSORT if you do not want Application Server to presort, but instead have the RDBMS sort itself when creating an index.</p>
NTHREADS <number>	<p>Allows you to run two processes at the same time when <number> is greater than 1. NTHREADS 2 allows one thread to carry out the Application Server and sgtrans processing and another thread to carry out the RDBMS processing concurrently.</p> <p>If you omit the NTHREADS keyword, or if you specify NTHREADS 1, only one process will occur at a time.</p> <p>For example, during an EXPORT DIMENSION or EXPORT DATA command, Application Server is reading dimensions members or time series respectively, collecting the information into batches of rows, and then sending a batch to the RDBMS for insertion into a dimension or fact table.</p> <p>In a single thread, Application Server has to wait while the RDBMS does the inserts before it prepares the next batch. If NTHREADS is set to 2 and two threads are running, Application Server can be preparing the next batch of rows to give to the RDBMS while the RDBMS is inserting the last batch of rows.</p> <p>Tip: On multiprocessor systems, using multiple threads concurrently should always improve throughput during reading and writing processes. On single processor systems, it may also improve throughput if a chunk of any elapsed time is I/O during which the CPU would otherwise be idle.</p>
TABLESPACE <tablespace_name>	Create the tables on this tablespace (Oracle or DB2). Used with EXPORT DIMENSION, TABLESPACE <tablespace_name> allows you to distribute tables across many disks.
USENAMES	Uses short names instead of codes in fact tables. The USENAMES keyword sets D_USENAMES in the prefix_DIMENSIONS table.
CODETYPE <datatype>	Specify the RDBMS datatype of the dimension column to control how to store codes in fact tables:

SMALLINT	2 bytes	A short integer between -32,768 and 32,767.
INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.
NUMBER(<p>,<s>)		Packed decimal fixed point data with <p> digits, <s> of which are to the right of the decimal point. Specify either NUMBER(<p>,<s>), NUMERIC(<p>,<s>), DECIMAL(<p>,<s>), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.
CHAR(<n>)		A fixed length character string using <n> bytes.
VARCHAR(<n>)		A varying length character string using up to <n> bytes.

Notes:

- The CODETYPE keyword sets the D_DATATYPE in prefix_DIMENSIONS.
- Oracle stores any numeric datatypes in packed decimal varying length data internally so SMALLINT, INTEGER, NUMBER(p,s), REAL, and FLOAT are all the same. If using DB2, all the datatypes apply.

Remarks

When you create a schema manually from the IDQL command line in Application Server, you must run the EXPORT BASE command before you can export dimensions. The EXPORT DIMENSION command creates the following relational tables in the schema for each dimension specified:

Dimension Levels table

Dimension tables

Notes:

In Hybrid OLAP models, dimension names should be of no more than 17 characters.

While using member names instead of codes makes the fact tables more readable, textual names in VARCHAR columns will normally take up more space than INTEGER columns or SMALLINT columns. Using the correct sized integral datatypes will usually save space and improve performance in all aspects.

8.12 EXPORT MASTER (Schema)

Use

EXPORT MASTER is a Hybrid OLAP SCHEMA subsystem command. EXPORT MASTER exports the Master table and creates and populates the Table Parameters table with entries for all of the other base tables, Dimension tables, the Variables table, and the Tracker table. You can amend the entries in the Table Parameters table before performing any other Hybrid OLAP EXPORT commands, thereby controlling the size, placement, and privileges on these tables.

Syntax

EXPORT MASTER

```
[ARRAYSIZE <number>]
[INDEXSPACE <tablespace_name>]
[TABLESPACE <tablespace_name>]
```

Parameter	Description
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.

EXPORT SCHEMA (Schema)

INDEXSPACE <tablespace_name>

Create the indexes on this tablespace (Oracle or DB2). Used with EXPORT MASTER, INDEXSPACE <tablespace_name> allows you to distribute indexes across many disks.

TABLESPACE <tablespace_name>

Create the tables on this tablespace (Oracle or DB2). Used with EXPORT MASTER, TABLESPACE <tablespace_name> allows you to distribute tables across many disks.

Remarks

You can use the Table Parameters table to control placement of all schema tables except for the Master table and the Table Parameters table itself. Placement of the Master table and Table Parameters table can be controlled via the TABLESPACE and INDEXSPACE keywords on the EXPORT BASE command.

If either the EXPORT BASE or EXPORT SCHEMA command is issued without a prior EXPORT MASTER command, the Table Parameters table will still be created, and will be populated with as many reasonable values as possible. However, explicit DBA control over many elements of the objects created during the schema export process is lost.

Note: Although they have no Table Parameters table, schemas created with earlier versions of Hybrid OLAP will continue to work. However, you will not have the same degree of control over table size, placement, and privileges for new tables or indexes.

8.13 EXPORT SCHEMA (Schema)

Use

EXPORT SCHEMA is a Hybrid OLAP SCHEMA subsystem command. EXPORT SCHEMA creates base tables in the schema, exports all dimension and variable information, and optionally exports data from an Application Server database.

Syntax

EXPORT SCHEMA

```
[ARRAYSIZE <number>]
[DATA [COMMIT [EVERY <rows>]]]
[INDEXSPACE <tablespace_name>]
[NOANALYZE]
[NOFOREIGNKEYS]
[NOSORT]
[NTHREADS <number>]
[TABLESPACE <tablespace_name>]
{ TIME ACROSS [ V_CODE <text> ] [USenames ] |
  TIME DOWN <periodicity> <dateformat> NAME <colname> TIMETYPE <datatype> }
[UNRECOVERABLE]
```

Parameter	Description
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.
DATA	Exports all data from the Application Server database to the schema.
COMMIT	Determines that a SQL COMMIT command should be executed after a specified number of data rows have been read.
EVERY	May be added after the COMMIT keyword to improve readability.
<rows>	The number or rows after which a SQL COMMIT command should be executed when reading data. The default is 250 rows.

INDEXSPACE <tablespace_name>	<p>Create the indexes on this tablespace (Oracle or DB2). Used with EXPORT SCHEMA, INDEXSPACE <tablespace_name> allows you to distribute indexes across many disks.</p> <p>Note: The schema base tables are ignored by INDEXSPACE <tablespace_name>.</p>
NOANALYZE	<p>For Oracle, the NOANALYZE keyword omits running the ANALYZE command to gather statistics for the cost-based optimizer. For DB2, the NOANALYZE keyword omits running the RUNSTATS command to gather statistics for the cost based optimizer. Note: Use this option with care. It is important that the statistics on tables be kept up to date, since this is the basis on which the RDBMS decides how to execute queries. If statistics are not available, or out of date, this may lead to the RDBMS choosing poor execution plans for queries leading to poor performance.</p>
NOFOREIGNKEYS	Foreign key integrity constraints are not created.
NOSORT	<p>(Oracle) Normally Application Server will write out data to the RDBMS in the order of any primary key in the table. By presenting data presorted to the RDBMS, index creation is faster in the RDBMS because the RDBMS does not have to presort data itself.</p> <p>Specify NOSORT if you do not want Application Server to presort, but instead have the RDBMS sort itself when creating an index.</p>
NTHREADS <number>	<p>Allows you to run two processes at the same time when <number> is greater than 1. NTHREADS 2 allows one thread to carry out the Application Server and sgtrans processing and another thread to carry out the RDBMS processing concurrently.</p> <p>If you omit the NTHREADS keyword, or if you specify NTHREADS 1, only one process will occur at a time.</p> <p>For example, during an EXPORT DIMENSION or EXPORT DATA command, Application Server is reading dimensions members or time series respectively, collecting the information into batches of rows, and then sending a batch to the RDBMS for insertion into a dimension or fact table.</p> <p>In a single thread, Application Server has to wait while the RDBMS does the inserts before it prepares the next batch. If NTHREADS is set to 2 and two threads are running, Application Server can be preparing the next batch of rows to give to the RDBMS while the RDBMS is inserting the last batch of rows.</p> <p>Tip: On multiprocessor systems, using multiple threads concurrently should always improve throughput during reading and writing processes. On single processor systems, it may also improve throughput if a chunk of any elapsed time is I/O during which the CPU would otherwise be idle.</p>
TABLESPACE <tablespace_name>	<p>Create the tables on this tablespace (Oracle or DB2). Used with EXPORT SCHEMA, TABLESPACE <tablespace_name> allows you to distribute tables across many disks.</p> <p>Note: The schema base tables are ignored by TABLESPACE <tablespace_name>.</p>
TIME ACROSS	<p>Specifies that the schema will be exported with time going across the fact tables. This is the default setting.</p> <p>Tip: Store time going down the table if you want to optimize the time it takes to load and consolidate data. Store time going across the table if you want to optimize end-user performance.</p>
V_CODE <text>	Uses the value of <text> as the column name for variables in fact tables instead of the word V_CODE when time is stored across the table.
USENAMES	Uses short names instead of codes in fact tables.
TIME DOWN	<p>Specifies that the schema will be exported with time going down the fact tables. Specify a repeating clause of <periodicity> <dateformat> NAME <colname></p>

EXPORT SCHEMA (Schema)

	TIMETYPE <datatype>, one for each different periodicity that is going to be present in fact tables. When you specify TIME DOWN, a new line is entered in the prefix_MASTER file that looks like this:		
	MST_KEY	MST_SETTING	
	TIME_ORIENTATION	DOWN	
<periodicity>	One of the following:		
	Yearly	12 months	
	Semiannual	6 months	
	Quarterly	quarter	
	Bimonthly	2 months	
	Monthly	month	
	Lunar	28 days. Use this with a 13-month fiscal year only.	
	Weekly	week	
	Biweekly	2 weeks	
	Daily	day	
<dateformat>	Type of data the field contains:		
<text>	Specifies that the TIME column will be character columns. Supplying a text string of XXX for example, means that the TIME column will be character column with values XXX1, XXX2, XXX3. The number is the period number (relative to the earliest period).		
ORDINAL	Specifies that you just want the relative period number without any text prefix in the TIME column. This way, the TIME column contains just the period number, for example 1, 2, 3.		
<date>	One of these: myy , ymd , yymd , dmyy , dmy , mdyy , mdy , ydm , yydm , yy , my , or ym . Specifies that the columns will be in date format. If the year is four digits (2002), use a format with yy . If the year is two digits (02), use a format with y . If the month is alphabetic (May 2000), replace the letter m with the word month , for example monthyy . Or specify Month/d/y for a date of Dec/3/02. Note: Oracle supports character translation with month names in its TO_DATE function so it is acceptable to use month names if using Oracle. If you are using an RDBMS other than Oracle, you should avoid using month names if possible. No other RDBMS supports character translation; using month names will cause the SQL to become complex and most likely perform poorly. You can use the separator characters dash (-), slash(/), or period (.) in the date format where appropriate. For example, specify m-d-y for a date of 12-3-02 and specify m/d/y for a date of 12/3/02.		
NAME <colname>	Specify the time column name for that periodicity.		
TIMETYPE <datatype>	Specify the RDBMS datatype of the time column:		
	SMALLINT	2 bytes	A short integer between -32,768 and 32,767.
	INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.
	REAL	4 bytes	A single-precision floating-point value with a range of 3.402823E38 to 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.
	FLOAT	8 bytes	A double-precision floating-point value with a range of 1.79769313486232E308 to 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.
	NUMBER(p,s)	Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.	

CHAR(n)	a fixed length character string using n bytes.
VARCHAR(n)	a varying length character string using up to n bytes.
DATE	A date string stored in the internal date format of the RDBMS.
DATETIME	A date and time string stored in the internal date format of the RDBMS.

Note: Oracle stores any numeric datatypes in packed decimal varying length data internally so SMALLINT, INTEGER, NUMBER(p,s), REAL, and FLOAT are all the same. If using DB2, all the datatypes apply.

UNRECOVERABLE (Oracle) Create all indexes without line redo log writes.

Remarks

You can use the EXPORT SCHEMA command to run the EXPORT BASE, EXPORT DIMENSION, and EXPORT VARIABLE commands in a single step. Using the DATA keyword is the same as executing the DUMP command from the Application Server database to the schema.

Note: If you already have fact tables, the datatypes and date formats should be set to match those which exist in those tables. Hybrid OLAP handles the information appropriately. If you are creating fact tables for the first time and you want to store time information going down the column, it is important to consider the appropriate datatypes and date format to use. You should choose a datatype and date format that leads to the simplest and most efficient SQL. For example choose an INTEGER datatype with a format YYMD or DATE or DATETIME. These datatypes result in simple SQL, which the RDBMS handles well. If you have date values in character columns with optional separators, when HOLAP has to query fact tables for data between two given dates, the SQL must use the RDBMS functions to translate the character strings to dates, which is extra work. It may also mean that the RDBMS cannot use an index on the fact table to its best advantage.

8.14 EXPORT SKELETON (Schema)

Use

EXPORT SKELETON is a Hybrid OLAP SCHEMA subsystem command. EXPORT SKELETON creates a skeleton schema without exporting any actual dimension, variable, or data information from Application Server.

Syntax

```
EXPORT SKELETON DIMENSION <dimensions> LEVELS <levels> OBSERVATIONS
<observations>[ARRAYSIZE <number>]
```

Parameter	Description
DIMENSION <dimensions>	One or more dimension names separated by commas. In Hybrid OLAP models, dimension names should be of no more than 17 characters.
LEVELS <levels>	Determines the dimension levels that should be created in the schema, based on a specified dimension level list.
OBSERVATIONS <observations>	Determines the number of observations that you expect to maintain in the schema, based on a specified observations list.
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.

Remarks

The settings you specify when you export a skeleton schema do not have to match the dimension and variable settings in the current Application Server database. This means that you can use the EXPORT SKELETON command to design a schema to your own requirements.

EXPORT VARIABLE (Schema)

After you have run the EXPORT SKELETON command, you can create a new Application Server database and then use the IMPORT SCHEMA command to map the Application Server database to the schema.

8.15 EXPORT VARIABLE (Schema)

Use

EXPORT VARIABLE is a Hybrid OLAP SCHEMA subsystem command. EXPORT VARIABLE exports information about the specified variables in an Application Server database to the schema.

You export the variables individually when using an EXPORT BASE command to export the model and you want to provide a different datatype for each variable.

Syntax

EXPORT VARIABLE <variables> [ALIAS <name>] [ARRAYSIZE <number>] [DATATYPE <datatype>]

Parameter	Description															
<variables>	One or more variable names separated by commas. You can specify an asterisk character (*) to export all variables defined in an Application Server database.															
ALIAS <name>	Uses the value for <name> as the fact column name rather than the variable name. Tip: By adding column names that are familiar to you in the fact tables, it is much easier to navigate through the fact tables in an ad hoc manner using SQL code.															
ARRAYSIZE <number>	Specifies a number between 1 and 32,767 for the size of the array you want to transmit to and from the RDBMS during the consolidation process. If you do not specify an arraysize, rows are transmitted to and from the RDBMS in batches of 1,000.															
DATATYPE <datatype>	(Available only when you previously used the EXPORT BASE TIME DOWN command) Specifies the RDBMS datatype of the variable column in which the data is stored when time is stored down the column in fact tables. You can specify a datatype for variables only when you have initially used the EXPORT BASE command with the TIME DOWN keywords. <table><tr><td>SMALLINT</td><td>2 bytes</td><td>A short integer between -32,768 and 32,767.</td></tr><tr><td>INTEGER</td><td>4 bytes</td><td>A long integer between -2,147,483,648 and 2,147,483,647.</td></tr><tr><td>REAL</td><td>4 bytes</td><td>A single-precision floating-point value with a range of 3.402823E38 to 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.</td></tr><tr><td>FLOAT</td><td>8 bytes</td><td>A double-precision floating-point value with a range of 1.79769313486232E308 to 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.</td></tr><tr><td>NUMBER(p,s)</td><td colspan="2">Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.</td></tr></table>	SMALLINT	2 bytes	A short integer between -32,768 and 32,767.	INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.	REAL	4 bytes	A single-precision floating-point value with a range of 3.402823E38 to 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.	FLOAT	8 bytes	A double-precision floating-point value with a range of 1.79769313486232E308 to 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.	NUMBER(p,s)	Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.	
SMALLINT	2 bytes	A short integer between -32,768 and 32,767.														
INTEGER	4 bytes	A long integer between -2,147,483,648 and 2,147,483,647.														
REAL	4 bytes	A single-precision floating-point value with a range of 3.402823E38 to 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.														
FLOAT	8 bytes	A double-precision floating-point value with a range of 1.79769313486232E308 to 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.														
NUMBER(p,s)	Packed decimal fixed point data with s digits, p of which are to the right of the decimal point. Specify either NUMBER(p,s), NUMERIC(p,s), DECIMAL(p,s), or DEC(p,s). Hybrid OLAP accepts any keyword as the same and generates the appropriate data type for the target RDBMS.															

Notes:

By default, the datatype is derived from the Application Server type. For example, if Sales is INTEGRAL BYTES 2 in Application Server, then EXPORT VARIABLE will put SMALLINT in V_DATATYPE. If Sales in Application Server is NU4, EXPORT VARIABLE will put REAL in V_DATATYPE. If you specify a DATATYPE, this value overrides the one set in Application Server for this variable.

This value overrides the default written to the prefix_VARIABLES column V_DATATYPE to specify the data type to store period data in fact tables.

Oracle stores any numeric datatypes in packed decimal varying length data internally so SMALLINT, INTEGER, NUMBER(p,s), REAL, FLOAT are all the same. If using DB2, all the datatypes apply.

Remarks

When you create a schema manually from the IDQL command line in Application Server, you must run the EXPORT BASE command before you can export variables. The EXPORT VARIABLE command inserts data into the following relational tables in the schema for each variable specified:

By Dimensions table

Format table

Variables table

Notes:

While using member names instead of codes makes the fact tables more readable, textual names in VARCHAR columns will normally take up more space than INTEGER columns or SMALLINT columns. Using the correct sized integral datatypes will usually save space and improve performance in all aspects.

8.16 IMPORT DATA (Schema)

Syntax

```
IMPORT DATA [* | <variable> | SELECTED] [PROGRESS <number>]
```

Use

IMPORT DATA is a Hybrid OLAP SCHEMA subsystem command that imports data defined in SAP NetWeaver BI into an existing Application Server database.

Internally IMPORT DATA imports data for one variable at a time and generates an MDX statement for each one. IMPORT DATA only imports INPUT data (transaction data). If any single variable hits the 1 million limit barrier, the data for that variable will not be imported.

IMPORT DATA fetches up to 1 million cells in any one MDX query. Importing BI cube data directly into an Application Server database is best used on small cubes. It is recommended for quick display times.

If you want to display SAP NetWeaver BI data from larger InfoCubes, you must use the IMPORT DIMENSION, IMPORT TIME, IMPORT VARIABLES, and IMPORT QUERY VARIABLES commands to access the cube data without actually importing it into Application Server. By accessing the data without importing it, you retrieve and cache information about the cubes in the Application Server database while the data remains in the BI cube.

See the *Server Configuration Guide for SAP BusinessObjects Strategy Management* located on SAP Service Marketplace to learn how to use this command.

Parameter	Description
*	You can also specify an asterisk character (*) to import all data defined in the schema.
<variable>	Name of variable to import.
SELECTED	Imports input data from the selected dimensions.
PROGRESS <number>	By default, the IMPORT DATA command gives reassurance messages at every 10,000 rows read from the provider. Use PROGRESS <number> to change the number of rows processed before getting a reassurance message. You can turn off reassurance messages by setting PROGRESS 0 or by issuing a SET REASSURANCE OFF command.

8.17 IMPORT DIMENSION (Schema)

Use

IMPORT DIMENSION is a Hybrid OLAP SCHEMA subsystem command. IMPORT DIMENSION imports dimensions defined in a schema into an existing Application Server database.

Syntax

```
IMPORT DIMENSION <dimensions>
  [REGULAR | STRUCTURED | UNSTRUCTURED | ATTRIBUTE]
  * | <names>
  [PREFACE <literal>]
```

Parameter	Description
<dimensions>	One or more dimension names separated by commas. You can also specify an asterisk character (*) to import all dimensions defined in the schema.
REGULAR	Imports dimensions that are not attribute or hierarchical attribute dimensions.
STRUCTURED	Imports only hierarchical attribute dimensions.
UNSTRUCTURED	Imports only non-hierarchical attribute dimensions.
ATTRIBUTE	Imports all attribute dimensions, regardless of whether they are hierarchical.
* or <names>	Imports all dimensions of the specified type, or a single named dimension.
PREFACE "<literal>"	<p>Prefaces the resultant dimension with the specified option. You must enclose the keywords in double quotation marks (" "). This option, followed by a quoted string, allows you to put anything in either a dimension or an attribute set.</p> <p>Valid options include: [KEY BOTH] [ALLOCATE [UPDATE] <input>, <output>, <result>]</p> <p>For example, specifying the command: IMPORT DIMENSION PRODUCT PREFACE "ALLOCATE UPDATE 50,10,1"</p> <p>generates an ALLOCATE statement so that when you import the PRODUCT dimension, the database is not reorganized when the dimension changes.</p> <p>Additionally, within the literal string, you can specify FACTOR <i>n</i>Percent, where <i>n</i>Percent is a percentage growth factor to use where there is no ALLOCATE statement in the PREFACE string. If you do not specify a PREFACE string, or the PREFACE string does not include a FACTOR statement, a default factor of 25% is used.</p>

Remarks

When you create dimensions in an Application Server database using the IMPORT DIMENSION command, the dimensions are automatically created and compiled. Any existing dimensions in the Application Server database that have the same name as the dimension being imported will be completely overwritten.

See the *Server Configuration Guide for SAP BusinessObjects Strategy Management* located on SAP Service Marketplace to learn how to use this command.

8.18 IMPORT DIMENSION (SAP NetWeaver BI Connector)

Use

IMPORT DIMENSION is a Hybrid OLAP SCHEMA subsystem command. IMPORT DIMENSION imports dimensions defined in a schema into an existing Application Server database.

The IMPORT DIMENSION command will retrieve the meta data for BI characteristics from the target cube and create corresponding dimensions in Application Server.

See the *Server Configuration Guide for SAP BusinessObjects Strategy Management* located on SAP Service Marketplace to learn how to use this command.

Syntax

```
IMPORT DIMENSION { * | { <dimension> [, <dimension>... ] } [INCLUDING FISCAL] [RANGE
<date1>-<date2>] [HIERARCHIES {ALL|DEFAULT}] [FORCE]
```

Parameter	Description
*	Imports all dimensions defined in the schema. Use * if the raw InfoCube or Query Cube has at most 12 dimensions excluding Time and excluding Navigational Attributes. You can use * if you created a Query Cube to eliminate some Characteristics if there were too many for Application Server. If you use the simple IMPORT DIMENSION * command then the IMPORT will automatically figure out what Navigational Attributes there are to import.
<dimension>	One or more dimension names separated by commas. If you have a raw InfoCube that has more than 12 Characteristics, you could issue individual IMPORT DIMENSION commands to import just the dimensions you want to use for analysis from the cube (and IMPORT at most 12 of those). Since in ODBO and OLAP BAPI the Navigational Attributes appear as Characteristics, you import Navigational Attributes into PAS Attributes this way too. If you IMPORT them one at a time you should import the Characteristic first and the related Navigational Attributes afterwards.
INCLUDING FISCAL	Includes fiscal information.
RANGE <date1>-<date2>	Specifies the date range to import from a Time characteristic. Use this as an alternative to restricting unwanted dates from a Query cube by using restrictions in BEx Query Designer. If you do not supply any RANGE then Application Server implicitly works as if you had done a RANGE 1/1/1900-12/31/2099 and will ignore any date outside the years 1900 through 2099.
SPANS DERIVED	Uses the values in the Time characteristic. This is the default setting. Allows you to specify how to determine the span of Measures in Application Server (i.e. the span you see when you do a SHOW VARIABLES in Application Server). The quickest way for Application Server to determine each Measure's span is to use the span it sees in the master data entries for the Time characteristic it uses as the basis for the Time component in Application Server on the IMPORT TIME command. SPANS DERIVED gives the fastest IMPORT time. If the dates have not been restricted, it is common in a Time dimension to have a far greater span defined in the Time characteristic (ready for future data loads and to define rollups for future dates) than actually exists. Using the values seen in the Time characteristic would result in Application Server thinking that Measures had a much bigger span of data than they really do and causing it to allocate and use unnecessarily large arrays for time series. In this case, it would be better to have Application Server actually use MDX to query the real time span of Measures or Key Figures. This takes longer (particularly if there are many measures) but gives exact results for the Time spans.
SPANS EXACT	Executes MDX for each measure to determine the exact spans.
HIERARCHIES ALL	Imports all hierarchies. If you omit the HIERARCHIES keyword, then HIERARCHIES ALL is the default setting and all hierarchies are imported.

IMPORT SCHEMA (Schema)

HIERARCHIES DEFAULT Imports the default BI hierarchy from each BI characteristic. In some cases limiting the hierarchies to the default and not importing all hierarchies can make the IMPORT command run much faster and produce much smaller Application Server databases.

FORCE Forces an import even if the CUBES timestamps identifies that nothing has changed. Use FORCE when you think the caches in Application Server are out-of-date or incorrect.

8.19 IMPORT SCHEMA (Schema)

Use

IMPORT SCHEMA is a Hybrid OLAP SCHEMA subsystem command. IMPORT SCHEMA imports variables and dimensions defined in a schema into an existing Application Server database.

Syntax

IMPORT SCHEMA

Remarks

Using IMPORT SCHEMA is the equivalent of executing IMPORT FISCAL, IMPORT DIMENSION, and IMPORT VARIABLES for all variables and dimensions defined in the schema. This makes it easy to populate an existing Application Server database from any schema that conforms to the Application Server Hybrid OLAP specifications.

Note: When a fiscal calendar is already set and variable data exists, IMPORT SCHEMA will not execute the IMPORT FISCAL command. Instead, it will use the existing fiscal calendar.

8.20 IMPORT SCHEMA (SAP NetWeaver BI Connector)

Use for SAP NetWeaver BI Connector

IMPORT SCHEMA is a Hybrid OLAP SCHEMA subsystem command. IMPORT SCHEMA imports variables and dimensions defined in a schema into an existing Application Server database.

If the source InfoCube or BEx Query Cube has at most 12 Characteristics (excluding Time and Navigational Attributes), and there is only one Time Characteristic, then you can map all the meta data into Application Server in one simple step using IMPORT SCHEMA.

The IMPORT command caches meta data information from SAP NetWeaver BI in Application Server. In BAPI and ODBO, the CUBES rowset provides timestamp information including when the cube was created, when the cube was last updated (eg a new characteristic was added or a Query was changed), and when the cube last had new data loaded.

See the *Server Configuration Guide for SAP BusinessObjects Strategy Management* located on SAP Service Marketplace to learn how to use this command.

Syntax

IMPORT SCHEMA [INCLUDING FISCAL] [RANGE <date1>-<date2>] [SPANS {DERIVED | EXACT}] [HIERARCHIES {ALL | DEFAULT}] [SYSVAR] [CHARACTERISTIC {'0CALDAY'|'0CALMONTH'|'0CALQUARTER'|'0CALWEEK'|'0CALYEAR'}] [FORCE]

Parameter	Description
INCLUDING FISCAL	Includes fiscal information.
RANGE <date1>-<date2>	Specifies the date range to import from a Time characteristic. Use this as an alternative to restricting unwanted dates from a Query cube by using

	restrictions in BEx Query Designer. If you do not supply any RANGE then Application Server implicitly works as if you had done a RANGE 1/1/1900-12/31/2099 and will ignore any date outside the years 1900 through 2099.
SPANS DERIVED	<p>Uses the values in the Time characteristic. This is the default setting.</p> <p>Allows you to specify how to determine the span of Measures in Application Server (i.e. the span you see when you do a SHOW VARIABLES in Application Server). The quickest way for Application Server to determine each Measure's span is to use the span it sees in the master data entries for the Time characteristic it uses as the basis for the Time component in Application Server on the IMPORT TIME command. SPANS DERIVED gives the fastest IMPORT time.</p> <p>If the dates have not been restricted, it is common in a Time dimension to have a far greater span defined in the Time characteristic (ready for future data loads and to define rollups for future dates) than actually exists. Using the values seen in the Time characteristic would result in Application Server thinking that Measures had a much bigger span of data that they really do and causing it to allocate and use unnecessarily large arrays for time series. In this case, it would be better to have Application Server actually use MDX to query the real time span of Measures or Key Figures. This takes longer (particularly if there are many measures) but gives exact results for the Time spans.</p>
SPANS EXACT	Executes MDX for each measure to determine the exact spans.
HIERARCHIES ALL	Imports all hierarchies. If you omit the HIERARCHIES keyword, then HIERARCHIES ALL is the default setting and all hierarchies are imported.
HIERARCHIES DEFAULT	Imports the default BI hierarchy from each BI characteristic. In some cases limiting the hierarchies to the default and not importing all hierarchies can make the IMPORT command run much faster and produce much smaller Application Server databases.
SYSVAR	Specifies that the Time characteristic in the Query contains System Variables (with or without offsets) and refreshes the Time cache based on the actual current contents of the Time dimension members.
CHARACTERISTIC	<p>Specify CHARACTERISTIC with a keyword to specify which Time characteristic to use. Application Server only expects one Time characteristic and it must be able to provide a date.</p> <p>0CALDAY is good for data of any periodicity.</p> <p>0CALMONTH is good for monthly data</p> <p>0CALQUARTER is good for quarterly data</p> <p>0CALWEEK is good for weekly data</p> <p>0CALYEAR is good for yearly data</p>
FORCE	Forces an import even if the CUBES timestamps identifies that nothing has changed. Use FORCE when you think the caches in Application Server are out-of-date or incorrect. Existing dimension caches in Application Server will be overwritten with new dimension caches. Existing SAP Query Variables information will be updated. The cached Time information will be updated. Any new Measures or Key Figures in the Cube will be added to Application Server and any time spans of Measures that had already been imported into Application Server will be updated.

Remarks

Using IMPORT SCHEMA is the equivalent of executing the following:

IMPORT FISCAL

IMPORT DIMENSION

IMPORT TIME (SAP NetWeaver BI Connector)

IMPORT TIME

IMPORT VARIABLE *

IMPORT QUERY VARIABLES ... only for BEx Query Cubes

This makes it easy to populate an existing Application Server database from any schema that conforms to the Application Server Hybrid OLAP specifications.

Note: When a fiscal calendar is already set and variable data exists, IMPORT SCHEMA will not execute the IMPORT FISCAL command. Instead, it will use the existing fiscal calendar.

8.21 IMPORT TIME (SAP NetWeaver BI Connector)

Use for SAP NetWeaver BI Connector

IMPORT TIME is a Hybrid OLAP SCHEMA subsystem command. The IMPORT TIME command tells HOLAP which Time Characteristic to use. If you have created a Query Cube that has only one Time Characteristic, then you can just issue IMPORT TIME with no other CHARACTERISTIC keywords.

Syntax

```
IMPORT TIME [INCLUDING FISCAL] [RANGE <date1>-<date2>] [SPANS {DERIVED|EXACT}]
[SYSVAR] [CHARACTERISTIC {'0CALDAY'|'0CALMONTH'|'0CALQUARTER'|
'0CALWEEK'|'0CALYEAR'}] [FORCE]
```

Parameter	Description
INCLUDING FISCAL	Includes fiscal information.
RANGE <date1>-<date2>	Specifies the date range to import from a Time characteristic. Use this as an alternative to restricting unwanted dates from a Query cube by using restrictions in BEx Query Designer. If you do not supply any RANGE then Application Server implicitly works as if you had done a RANGE 1/1/1900-12/31/2099 and will ignore any date outside the years 1900 through 2099.
SPANS DERIVED	<p>Uses the values in the Time characteristic. This is the default setting.</p> <p>Allows you to specify how to determine the span of Measures in Application Server (i.e. the span you see when you do a SHOW VARIABLES in Application Server). The quickest way for Application Server to determine each Measure's span is to use the span it sees in the master data entries for the Time characteristic it uses as the basis for the Time component in Application Server on the IMPORT TIME command. SPANS DERIVED gives the fastest IMPORT time.</p> <p>If the dates have not been restricted, it is common in a Time dimension to have a far greater span defined in the Time characteristic (ready for future data loads and to define rollups for future dates) than actually exists. Using the values seen in the Time characteristic would result in Application Server thinking that Measures had a much bigger span of data than they really do and causing it to allocate and use unnecessarily large arrays for time series. In this case, it would be better to have Application Server actually use MDX to query the real time span of Measures or Key Figures. This takes longer (particularly if there are many measures) but gives exact results for the Time spans.</p>
SPANS EXACT	Executes MDX for each measure to determine the exact spans.
SYSVAR	Specifies that the Time characteristic in the Query contains System Variables (with or without offsets) and refreshes the Time cache based on the actual current contents of the Time dimension members.

CHARACTERISTIC	Specify CHARACTERISTIC with a keyword to specify which Time characteristic to use. Application Server only expects one Time characteristic and it must be able to provide a date. 0CALDAY is good for data of any periodicity. 0CALMONTH is good for monthly data 0CALQUARTER is good for quarterly data 0CALWEEK is good for weekly data 0CALYEAR is good for yearly data
FORCE	Forces an import even if the CUBES timestamps identifies that nothing has changed. Use FORCE when you think the caches in Application Server are out-of-date or incorrect.

Remarks

Using IMPORT SCHEMA is the equivalent of executing IMPORT FISCAL, IMPORT DIMENSION, and IMPORT VARIABLES for all variables and dimensions defined in the schema. This makes it easy to populate an existing Application Server database from any schema that conforms to the Application Server Hybrid OLAP specifications.

Note: When a fiscal calendar is already set and variable data exists, IMPORT SCHEMA will not execute the IMPORT FISCAL command. Instead, it will use the existing fiscal calendar.

8.22 IMPORT VARIABLES (Schema)

Use

IMPORT VARIABLES is a Hybrid OLAP SCHEMA subsystem command. IMPORT VARIABLES imports variables defined in a schema into an existing Application Server database.

Syntax

```
IMPORT VARIABLES { * | <variable> [,<variable>...]}
```

Parameter	Description
*	Imports all variables defined in the schema.
<variables>	One or more variable names separated by commas.

Remarks

When you create variables in an Application Server database using the IMPORT VARIABLES command, the variables are automatically created and assigned the Drillthru attribute. Any existing variables in the Application Server database that have the same name as the variables being imported need to be removed prior to issuing the IMPORT VARIABLES command.

8.23 IMPORT VARIABLES (SAP NetWeaver BI Connector)

Use

IMPORT VARIABLES is a Hybrid OLAP SCHEMA subsystem command. IMPORT VARIABLES imports variables defined in a schema into an existing Application Server database.

The IMPORT VARIABLE command imports information about Key Figures from SAP NetWeaver BI into PAS Variables (or Measures, Metrics). An IMPORT VARIABLE command will be able to determine the dimensionality and data type (INTEGRAL BYTES 1/2/4 or NUMERIC BYTES 4/8) and will create the variable with as much information as is available from SAP NetWeaver BI.

IMPORT QUERY VARIABLES (SAP NetWeaver BI Connector)

If you want to set properties like RATE, EXPENSE, UNITS, DECIMALS, WIDTH and whether the variable is to be time converted with SUM, FIRST, LAST, then following an IMPORT VARIABLES command you should issue SET VARIABLE commands.

Syntax

```
IMPORT VARIABLES { * | <variable> [, <variable> ...] } [FORCE]
```

Parameter	Description
*	Import all variables defined in the schema.
<variables>	One or more variable names separated by commas.

8.24 IMPORT QUERY VARIABLES (SAP NetWeaver BI Connector)

Use

The IMPORT QUERY VARIABLES command imports any query variable information from the SAP VARIABLES rowset for a Query Cube and caches it in Application Server.

You can examine the cache using the EXHIBIT QUERY VARIABLES command and you can set values for a Query Variable using the SET QUERY VARIABLE command.

Query Variables are placeholders in the Query that allow values to be supplied at run time rather than hard coding selections or values in the Query itself.

Query Variables can be placeholders for a member names in a dimension (ie a characteristic value in a characteristic) or a dimension hierarchy or for a numeric value that is used in a constraint or numeric value used in a formula in a calculated measure. Query Variables can be optional or mandatory and may or may not have a default value.

At run time, SAP NetWeaver BI has implemented extensions to MDX so that an application can supply values for Query Variables to the back end. The SAP NetWeaver BI Connector takes any values that you have set using the SET QUERY VARIABLE command and incorporates them into the generated MDX to pass them to the backend.

Syntax

```
IMPORT QUERY VARIABLES [FORCE]
```

Parameter	Description
FORCE	Forces an import even if the CUBES timestamps identifies that nothing has changed. Use FORCE when you think the caches in Application Server are out-of-date or incorrect.

8.25 PREFIX (Schema)

Use

PREFIX is a Hybrid OLAP SCHEMA subsystem command. PREFIX sets a prefix to be applied to all relational tables created as part of a schema

Syntax

```
PREFIX <prefix>
```

Parameter	Description
<prefix>	The prefix used in the name of the relational table. The default value is Application Server.

Remarks

Every relational schema that you create must have a prefix. The schema prefix must be of 5 or fewer bytes, and begin with an alphabetical character. If you have long dimension names (that is, dimension names of 10 or more characters, up to a maximum of 17), it is advisable to use a prefix shorter than 10 characters. The schema prefix is applied to all of the relational tables in a schema, and serves as a unique identifier.

An underscore character (_) is used to separate the prefix from the rest of the relational table name. For example, if you set the prefix to DEMO before creating a schema, the Dimensions table in the schema would be named DEMO_DIMENSIONS.

You can use a prefix to maintain more than one schema using the same Link source. For example, you could create two schemas in the same Oracle tablespace, one with the prefix DEMO1 and the other with the prefix DEMO2.

The defined prefix is saved in the Work database. This means that unless you execute the EXIT CLEAR command when you leave Application Server, the prefix will persist across sessions.

You can use the VIEW PREFIX command to display the currently defined prefix.

8.26 SCHEMA (Schema)

Use

SCHEMA is a Hybrid OLAP SCHEMA subsystem command. SCHEMA enters the SCHEMA subsystem.

Syntax

SCHEMA

Remarks

You use the SCHEMA command to enter the SCHEMA subsystem where you can enter Hybrid OLAP commands. Enable Hybrid OLAP before attempting to enter the SCHEMA subsystem.

Note: Once you have entered the SCHEMA subsystem, you will need to establish a Link connection to the relational database before you can perform Hybrid OLAP operations such as exporting an Application Server database to a schema. This can be done with the CONNECT command.

8.27 SET LABEL (Schema)

Use

SET LABEL is a Hybrid OLAP SCHEMA subsystem command. SET LABEL defines the label set to be used for the specified dimension or variable.

Syntax

SET LABEL <name> [DEFAULT | FROM <linkid> NAME <label> [LANGUAGE <language>] | SYNONYM <synonym>]

Parameter	Description
<name>	The name of the dimension or variable to which the labels are to be applied.
DEFAULT	Switches back to standard Application Server long and short names for the specified dimension or variable.
FROM <linkid>	Defines the Link ID used to connect to the schema.
NAME <label>	The name of the label set to be used for the specified dimension or variable, as defined in the corresponding Reference table.

SET VARIABLE (Schema)

- LANGUAGE <language> The name of the language set to be used for the specified dimension or variable, as defined in the corresponding Reference table.
- SYNONYM <synonym> The name of an Application Server synonym set containing alternative long and short names for the specified dimension or variable.

Remarks

Alternative names are stored in Reference tables in the relational schema. Reference tables must be created by the user or the RDBMS database administrator.

8.28 SET VARIABLE (Schema)

Use

SET VARIABLE is a Hybrid OLAP SCHEMA subsystem command. SET VARIABLE assigns the Drillthru attribute to Application Server variables. This means that data for the variables can be stored in relational tables.

Syntax

SET VARIABLE <variables> {FROM <linkid> PREFIX <prefix> | NOFROM}

Parameter	Description
<variables>	One or more variable names separated by commas. You can specify an asterisk character (*) to assign the Drillthru attribute to all variables.
FROM <linkid>	Defines the Link ID that should be used to connect to the schema.
PREFIX <prefix>	Defines the schema prefix.
NOFROM	Removes the Drillthru attribute from the specified variables.

8.29 SPY (Schema)

Use

SPY is a Hybrid OLAP SCHEMA subsystem command. SPY sends the SQL output generated by Hybrid OLAP to the Application Server output window, an internal set, or an external text file.

Syntax

SPY {[TERMINAL | <name> [:EXT]] | OFF}

Parameter	Description
TERMINAL	Displays the SQL output in the Application Server output window.
<name>	The name of an Application Server set to which SQL output will be sent.
EXT	Specifies that the Application Server set specified in <name> will be created external to the Application Server database.
OFF	Indicates that SQL output is no longer to be sent to the specified destination.

Remarks

You can use the SPY command to view the SQL code generated by Application Server with Hybrid OLAP when performing operations such as exporting an Application Server database to a new schema. This provides a useful insight into how Hybrid OLAP operates.

8.30 TRACKER INSERT (Schema)

Use

TRACKER INSERT is a Hybrid OLAP SCHEMA subsystem command. TRACKER INSERT populates any missing Tracker records in the Tracker table. Any existing records are left untouched.

Syntax

TRACKER INSERT {<variables> | SELECTED} [DIRECTLOADPATH]

Parameter	Description
<variables>	One or more variable names separated by commas. You can specify an asterisk (*) to populate the Tracker table with any missing records for all combinations for all variables.
SELECTED	Inserts the variable/dimension level combinations currently selected in the Application Server database.
DIRECTLOADPATH	Invokes the Oracle SQL*Loader and uses the DIRECT load path to load fact tables, create the primary key, and populate any other indexes at table creation time.

Remarks

Records inserted for input level combinations will be marked as base data in the RDBMS. Records inserted for output combinations will be marked as preconsolidated data in the RDBMS.

8.31 TRIGGER (Schema)

Use

TRIGGER is a Hybrid OLAP SCHEMA subsystem command. TRIGGER facilitates the management of database triggers on the tables in your schema.

Syntax

TRIGGER {DISABLE | ENABLE | DROP | REPLACE} {DIMENSION | FACT | BOTH} {*} <list of table names>}

Parameter	Description
DISABLE	Disables the trigger. For Oracle, this generates an ALTER TRIGGER ...DISABLE statement. The trigger is disabled, rather than dropped. The trigger is not executed upon subsequent INSERT, UPDATE, or DELETE statements until it is enabled again.
ENABLE	Re-enables the trigger. For Oracle, this generates an ALTER TRIGGER...ENABLE statement. Thereafter, the trigger will be fired again upon subsequent INSERT, UPDATE, or DELETE statements.
DROP	Physically drops the trigger. TRIGGER DROP may be useful during the prototyping stage of model development, or in cases where you will never do any incremental consolidation.
REPLACE	Replaces an existing trigger with an up-to-date version. Note: Triggers changed in version 6.1. Therefore, if you want to preserve existing schemas, you must manually issue the TRIGGER REPLACE BOTH * command when you upgrade from version 6.0.
DIMENSION	Apply the TRIGGER command only to Dimension table triggers.
FACT	Apply the TRIGGER command only to Fact table triggers.
BOTH	Apply the TRIGGER command to both Dimension and Fact table triggers.

VIEW CACHE (Schema)

- * Indicates that the TRIGGER command applies to all tables of the specified type.
- <list of table names> For Dimension table triggers, specify a list of table names. For Fact table triggers, specify a list of table names, including the prefix if necessary.

Remarks

To get the best performance out of initial data loads via the Transformer, disable triggers before the Transformer run, and re-enable them afterwards. That way, you avoid needlessly populating the Fact Delta table(s).

8.32 VIEW CACHE (Schema)

Use

VIEW CACHE is a Hybrid OLAP SCHEMA subsystem command. VIEW CACHE displays the current settings for the CACHE command.

Syntax

VIEW CACHE

8.33 VIEW CONNECTION (Schema)

Use

VIEW CONNECTION is a Hybrid OLAP SCHEMA subsystem command. VIEW CONNECTION displays a list of the Link IDs that are currently connected.

Syntax

VIEW CONNECTION

Remarks

The current active Hybrid OLAP connection is marked with an asterisk (*). A connection opened in the same session for other Hybrid OLAP access that is not the current Hybrid OLAP connection is marked with a plus sign (+). A connection opened with the Application Server ACCESS LSLINK command that is still open in this session is marked with neither symbol.

8.34 VIEW DIMENSION (Schema)

Use

VIEW DIMENSION is a Hybrid OLAP SCHEMA subsystem command. VIEW DIMENSION displays information about the dimensions in the current schema.

Syntax

VIEW DIMENSION <dimensions> [FULL]

Parameter	Description
<dimensions>	One or more dimension names separated by commas. You can specify an asterisk character (*) to display information about all dimensions defined in the schema.
FULL	Provides more detailed output.

8.35 VIEW PREFIX (Schema)

Use

VIEW PREFIX is a Hybrid OLAP SCHEMA subsystem command. VIEW PREFIX displays the current schema prefix, set using the PREFIX command.

Syntax

VIEW PREFIX

8.36 VIEW <rowsettype> ROWSET

Use

VIEW <rowsettype> ROWSET is a Hybrid OLAP SCHEMA subsystem command. VIEW SCHEMA is used to query BI meta data and produce tab-separated output suitable for copy and paste to Excel or any other grid utility. You should be familiar with MDX naming conventions to understand the output.

Syntax

VIEW {CATALOGS | CUBES | DIMENSIONS | MEASURES | HIERARCHIES | LEVELS | MEMBERS | PROPERTIES | QUERY VARIABLES} ROWSET [FULL]
[<Restriction>[,<Restriction>...]]

Parameter	Description
FULL	Provides more detailed output.
<restriction>	<restriction name> <restriction value>. Keywords and restriction names can be abbreviated to 3 characters.

Remarks

Each rowset also has a set of restriction columns (i.e. columns that you can specify values for to restrict the output to specific databases or cubes or dimensions). The following table shows the available restrictions for each rowset. SCHEMA_NAME is not supported in SAP NetWeaver BI and should not be used.

Rowset	Restrictions
CATALOGS	CATALOG_NAME
CUBES	CATALOG_NAME SCHEMA_NAME CUBE_NAME
DIMENSIONS	CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_NAME DIMENSION_UNIQUE_NAME
MEASURES	CATALOG_NAME SCHEMA_NAME CUBE_NAME MEASURE_NAME MEASURE_UNIQUE_NAME
HIERARCHIES	CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_NAME HIERARCHY_UNIQUE_NAME
LEVELS	CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_NAME LEVEL_UNIQUE_NAME
MEMBERS	CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_UNIQUE_NAME LEVEL_NUMBER MEMBER_NAME MEMBER_UNIQUE_NAME MEMBER_CAPTION

VIEW SPANS TYPE

	MEMBER_TYPE Tree operator
PROPERTIES	CATALOG_NAME SCHEMA_NAME CUBE_NAME DIMENSION_UNIQUE_NAME HIERARCHY_UNIQUE_NAME LEVEL_UNIQUE_NAME MEMBER_UNIQUE_NAME PROPERTY_NAME PROPERTY_TYPE
QUERY VARIABLES	CATALOG_NAME SCHEMA_NAME CUBE_NAME

A restriction is a single value that is compared for equality to the corresponding value in the schema rowset. Wildcards are not supported. Only rows that exactly match the restriction value in the corresponding column are returned in the rowset. So for example if you specify a restriction on CUBE_NAME being [\$0D_SD_C03] then only rows from the rowset that have the CUBE_NAME column having the value [\$0D_SD_C03] are returned. This is case sensitive in some ODBO providers like SAP NetWeaver BI and not in others (for MSAS it is not case sensitive). The values used in the restrictions must adhere to the ODBO naming conventions and you should enclose them in single quotes in the command line.

Many of these rowsets will output huge amounts of information if you do not use restrictions. For example the MEMBERS rowset will output every Member (i.e. a Characteristic Value) for every Dimension (i.e. Characteristic) for every Cube if you add no restrictions.

8.37 VIEW SPANS TYPE

Use

VIEW SPANS TYPE is a Hybrid OLAP SCHEMA subsystem command. VIEW SPANS TYPE displays the SPANS setting on IMPORT SCHEMA and IMPORT TIME.

Syntax

VIEW SPANS TYPE

8.38 VIEW TIME RANGE

Use

VIEW TIME RANGE is a Hybrid OLAP SCHEMA subsystem command. VIEW TIME RANGE displays the time range setting used to import the Cube.

Syntax

VIEW TIME RANGE

8.39 VIEW SCHEMA (Schema)

Use

VIEW SCHEMA is a Hybrid OLAP SCHEMA subsystem command. VIEW SCHEMA displays information about the variables and dimensions defined in the current schema.

Syntax

VIEW SCHEMA [FULL]

Parameter	Description
FULL	Provides more detailed output.

Remarks

VIEW SCHEMA is the equivalent of running VIEW VARIABLE and VIEW DIMENSION for all variables and dimensions defined in the schema.

8.40 VIEW SPY (Schema)

Use

VIEW SPY is a Hybrid OLAP SCHEMA subsystem command. VIEW SPY displays the current settings for the SPY command, including the destination of any generated SQL code.

Syntax

VIEW SPY

8.41 VIEW VARIABLE (Schema)

Use

VIEW VARIABLE is a Hybrid OLAP SCHEMA subsystem command. VIEW VARIABLE displays information about the variables in the current schema.

Syntax

VIEW VARIABLE <variables> [FULL]

Parameter	Description
<variables>	One or more variable names separated by commas. You can specify an asterisk character (*) to display information about all variables defined in the schema.
FULL	Provides more detailed output.

8.42 DROP DATA (Schema)

Use

DROP DATA is a Hybrid OLAP SCHEMA subsystem command. DROP DATA deletes the Tracker table and all Fact tables from the current schema.

Syntax

DROP DATA

Remarks

You should use the DROP REFERENCES command to drop all Reference tables from the schema before deleting the Tracker table and Fact tables from the schema. Alternatively, you can run the DROP SCHEMA command to delete the entire schema in a single step.

8.43 CONNECT (Schema)

Use

CONNECT is a Hybrid OLAP SCHEMA subsystem command. You must issue a CONNECT statement to access data stored in relational databases through a Link ID.

Syntax

CONNECT <linkid>

IMPORT FISCAL (Schema)

Parameter	Description
<linkid>	A valid Link ID. You create and maintain Link IDs from Explorer tab of Application Server.

8.44 IMPORT FISCAL (Schema)

Use

IMPORT FISCAL is a Hybrid OLAP SCHEMA subsystem command. IMPORT FISCAL imports the fiscal calendar information defined in a schema into an existing Application Server database.

Syntax

IMPORT FISCAL

Remarks

You can view the schema's fiscal calendar information in the Master table.

8.45 SQL (Schema)

Use

SQL is a Hybrid OLAP SCHEMA subsystem command. SQL allows you to issue SQL statements while you are in the Hybrid OLAP SCHEMA subsystem. The SQL statements that follow the SQL command are passed directly on to the RDBMS.

Syntax

SQL <SQL_statement>

Parameter	Description
<SQL_statement>	Any valid SQL statement except a SELECT.

Remarks

You must follow SQL commands with SQL COMMIT statements if they are required by your RDBMS.

8.46 ORDER DRILLTHRU (Schema)

Use

ORDER DRILLTHRU is a Schema subsystem command. ORDER DRILLTHRU allows you to order dimension members using a selection from the dimension in the schema, while ignoring any previously-made selections in Application Server.

Syntax

```
ORDER <dimension> ON <variable> [TOP <n> | BOTTOM <n>] SELECT DRILLTHRU
  { [ONLY] ABOVE <member> | BELOW <member> }
  { [JUST] ABOVE <member> | BELOW <member> }
  { [ONLY JUST] ABOVE <member> | BELOW <member> }
  [{MEMBER | LEVEL | INPUTS | OUTPUTS}]
  {AND LEVEL <n>}
```

Parameter	Description
<dimension>	Name of the dimension to order.
ON <variable>	Orders members from the member with the largest numeric value to the member with the smallest numeric value according to the specified expression.

TOP <n>	Orders just the top <n> selected members of the dimension.
BOTTOM <n>	Orders just the bottom <n> selected members of the dimension.
SELECT	Automatically selects just the members being ordered.
ABOVE	Selects the specified member and all the members in the path above it. If the dimension contains multiple hierarchies, <member> is a member in the currently set hierarchy. ABOVE selects the specified member and all the members in the same hierarchy above it.
<member>	Name of one or more members, separated by commas.
JUST ABOVE	Selects the specified member and the member in the path directly above it.
ONLY ABOVE	Selects all members in the path above the specified member, but not including the specified member.
ONLY JUST ABOVE	Selects only the member in the path directly above the specified member.
BELOW	Selects the specified member and all members in the path below it.
JUST BELOW	Selects the specified member and the members directly below it.
ONLY BELOW	Selects all members in the path below the specified member, but not including the specified member.
ONLY JUST BELOW	Selects only the members in the path directly below the specified member.
LEVEL	A level name or level number.

Remarks

Before including the DRILLTHRU keyword, the ORDER command [TOP n | BOTTOM n] keyword must be specified. In addition, when the DRILLTHRU keyword is used, the Application Server SELECT command becomes part of the ORDER DRILLTHRU statement, rather than coming before it on a separate line as in non-drillthrough models.

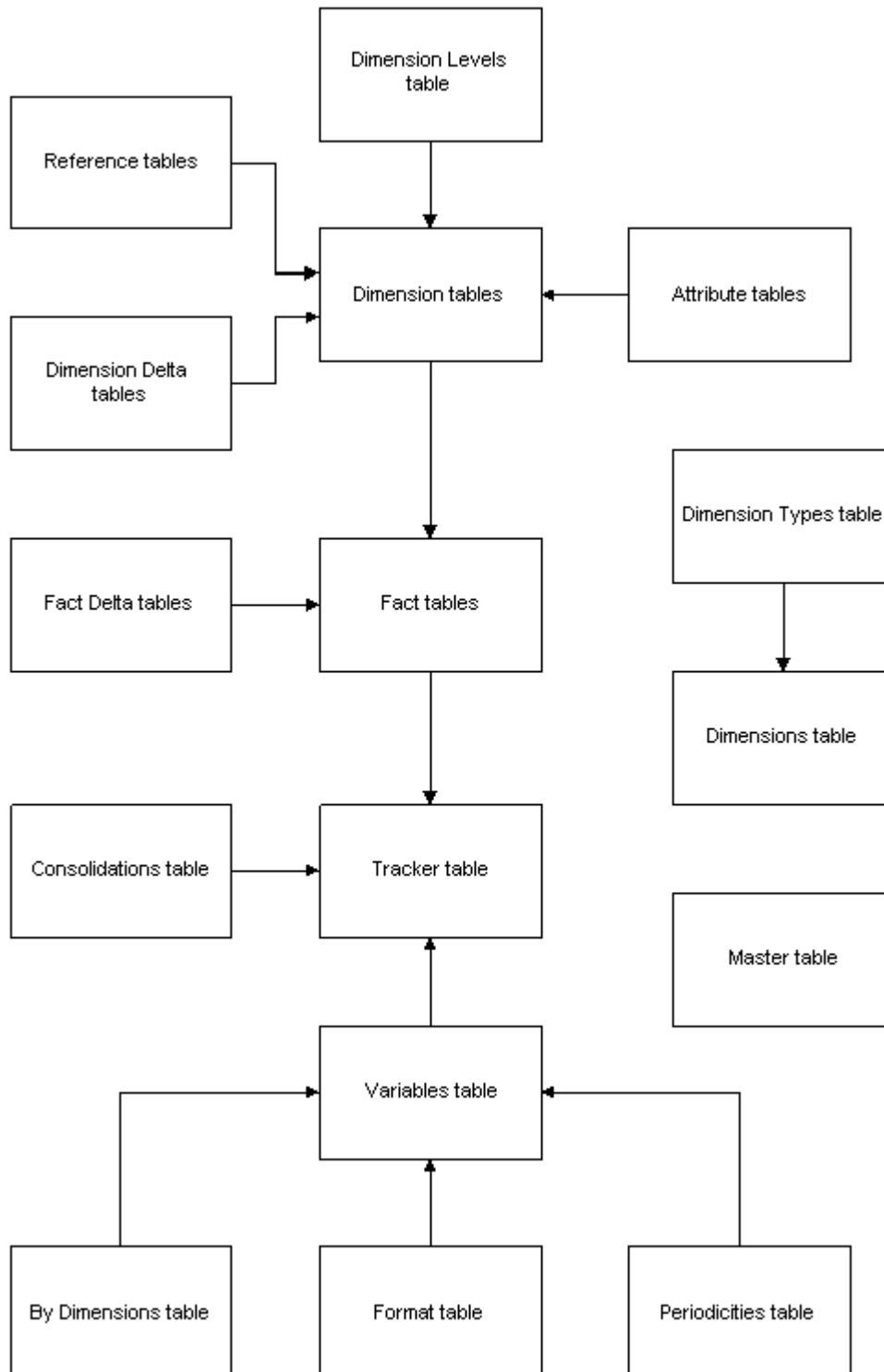
If there is nothing after DRILLTHRU, the whole dimension is used.

Note: See Application Server online Help command reference for a full description of the ORDER command.

9 Schema Table Reference

9.1 Schema Map

To see the entire schema map, you may need to scroll the Help window.



9.2 Consolidations Table

Use

The Consolidations table defines an identifier that determines the source of time series data to Application Server, and how that data should be consolidated.

Format

The Consolidations table is named *prefix_CONSOLIDATIONS*, where *prefix* is the name of the schema prefix. The Consolidations table consists of the following columns:

Column	Description
<i>cons_type</i>	The unique code for a specific consolidation type.
<i>cons_desc</i>	The description for a specific consolidation type.

Remarks

You use the Consolidations table to determine whether time series data should be stored in Application Server directly, or stored externally in the RDBMS, and whether data should be preconsolidated or consolidated dynamically.

Typically, you would store preconsolidated data values for the most frequently accessed time series data. Look at the NHITS column of the Tracker table to determine which data users access most frequently.

The *cons_type* column is used as a foreign key in the Tracker table, which determines where data is stored and how it should be consolidated. The contents of the Consolidations table are fixed, and will normally be generated by Application Server as part of the export process.

9.3 Fact Tables

Use

Fact tables are where the actual time series data is stored. Data for some time series may span across several Fact tables.

Format

Fact tables are named *prefix_FACT_extension*, where *prefix* is the name of the schema prefix, and *extension* is a system-generated value based on the variable data stored in the Fact table. Each Fact table consists of the following columns:

Column	Description
<i>v_code</i>	The unique code for a specific Application Server variable, as defined in the Variables table.
<i>m_dim</i>	Fact tables include a single column for each structural dimension in the Application Server database. These columns are named <i>m_dim</i> , where <i>dim</i> is the name of the dimension to which the column relates. For example, an Application Server database with dimensions Product, Region, and Type would have corresponding columns in each of its Fact tables named <i>m_product</i> , <i>m_region</i> , and <i>m_type</i> . The row values in each column represent a specific dimension member, as defined by the <i>m_code</i> for the member in the corresponding Dimension table.
<i>p1-pn</i>	These columns represent the individual time periods in Application Server. If data for a variable is stored monthly, each column represents data for a single month. The number of time period columns in a Fact table is defined by the corresponding <i>nobs</i> value in the Tracker table. The <i>start_date</i> value in the Tracker table defines the start date for data in the <i>p1</i> column in the Fact table.

Remarks

Each row in a Fact table represents the data for a variable and specific dimension member combination. For example, the Actual number of Units for 386sx Chips in Phoenix could be stored in a single row. The number of observations held for this data would determine whether the data could be stored in a single Fact table, or whether it had to be spanned over several Fact tables.

The export process will automatically create and populate the Tracker table and Fact tables and Application Server will have to make assumptions about how to store the data. Ideally, these default settings should be reviewed by the database administrator, to ensure that data is being stored in the most appropriate place, and that it is being consolidated in a way that will maximize performance.

9.4 Fact Delta Tables

Use

Fact Delta table(s) contain changes made to the Fact table(s).

Format

The Fact Delta tables are named *prefix_FACT_extension_D*, where *prefix* is the name of the schema prefix, and *extension* is a system-generated value based on the variable data stored in the Fact table. Each Fact Delta table consists of the following columns:

Column	Description
<i>v_code</i>	The unique code for a specific Application Server variable, as defined in the Variables table.
<i>m_dim</i>	The Fact tables include a single column for each structural dimension in the Application Server database. These columns are named <i>m_dim</i> , where <i>dim</i> is the name of the dimension to which the column relates. For example, an Application Server database with dimensions Product, Region, and Type would have corresponding columns in each of its Fact tables named <i>m_product</i> , <i>m_region</i> , and <i>m_type</i> . The row values in each column represent a specific dimension member, as defined by the <i>m_code</i> for the member in the corresponding Dimension table.
<i>p1-pn</i>	These columns represent the individual time periods in Application Server. If data for a variable is stored monthly, each column represents data for a single month. The number of time period columns in a Fact table is defined by the corresponding <i>nobs</i> value in the Tracker table. The <i>start_date</i> value in the Tracker table defines the start date for data in the <i>p1</i> column in the Fact table.

Remarks

There is one Fact Delta table for each Fact table in the schema. Fact Delta tables are not populated until changes are made.

9.5 Dimension Types Table

Use

The Dimension Types table defines whether an Application Server dimension is a structural dimension, structural attribute, or unstructured attribute.

Format

The Dimension Types table is named *prefix_DIM_TYPES*, where *prefix* is the name of the schema prefix. The Dimension Types table consists of the following columns:

Column	Description
<i>d_type</i>	The unique code for a specific Application Server dimension type.
<i>d_name</i>	The description for a specific Application Server dimension type.

Remarks

The *d_type* column becomes a foreign key in the Dimensions table. The contents of this table are fixed and will normally be generated by Application Server.

9.6 Dimensions Table

Use

The Dimensions table stores the names and types of Application Server dimensions.

Format

The Dimensions table is named *prefix_DIMENSIONS*, where *prefix* is the name of the schema prefix. The Dimensions table consists of the following columns:

Column	Description
<i>d_name</i>	The name of an Application Server structural or attribute dimension.
<i>d_type</i>	The unique code for a specific Application Server dimension type, as defined in the Dimension Types table.
<i>d_by_dim</i>	The name of the structural dimension to which an attribute applies. This column is only applicable when <i>d_name</i> specifies the name of an attribute dimension.
NO_GUESTS= <i>char</i>	<i>char</i> is any non-zero character. The user will not have the ability to do anything that will cause the dimension in Application Server to differ from the dimension in the schema. An IMPORT will always use the whole dimension. There will never be any guest members in the Application Server dimension. Application Server does not need to go to the RDBMS tables to retrieve dimension members. Application Server will ignore the MAX_GUEST_LEVEL. If NO_GUESTS is a character value of 0 or NULL, the dimension in Application Server is not guaranteed to be the whole dimension. There may be guest members. Application Server will take the MAX_GUEST_LEVEL setting into account.
MAX_GUEST_LEVEL= <i>n</i>	Specify a value for <i>n</i> . The dimension in Application Server is created via a view which did SELECT * FROM <i>prefix_dimname</i> WHERE M_LEVEL > <i>n</i> . Typically, the dimension in Application Server just excludes the inputs. In this case, Application Server can work out when it needs to go to the RDBMS and when it can do it without. If MAX_GUEST_LEVEL = NULL, the dimension in Application Server is not created from a view which just did SELECT * FROM <i>prefix_dimname</i> WHERE M_LEVEL > <i>n</i> . Members of a single level may be split between the RDBMS and Application Server. In this case Application Server will always have to check the RDBMS for dimension members. The EXPORT command will export a dimension and set NO_GUESTS = 1, MAX_GUEST_LEVEL = 0. The Transformer will determine how many rows are inserted into a dimension table. If there are fewer than 100K, it will make the settings just like the EXPORT command did. If there are more than 100K members, it will set NO_GUESTS = 0, MAX_GUEST_LEVEL = NULL. When the user creates views and imports dimensions, the MAX_GUEST_LEVEL setting should be updated based on the view to ensure the best performance when retrieving dimension members.

Remarks

The *d_type* column becomes a foreign key in the Dimension Types table. It is used by the IMPORT and EXPORT commands in Application Server to determine structural dimensions, structural attributes, and unstructured attributes. For structural attributes, the *d_by_dim* column specifies the

Dimension Tables

dimension that Application Server uses to create a BY DIMENSION clause. For unstructured attributes, *d_by_dim* specifies the dimension for which the attribute applies.

Note: In Hybrid OLAP models, dimension names should be of no more than 17 characters.

9.7 Dimension Tables

Use

The Dimension tables define the members of an Application Server dimension, and the hierarchical relationship between these members. There is a Dimension table for each Application Server structural dimension, together with a corresponding Dimension Levels table.

Format

The Dimension tables are named *prefix_dimension*, where *prefix* is the name of the schema prefix, and *dimension* is the name of an Application Server dimension. Each Dimension table consists of the following columns:

Column	Description
m_code	The unique code for an Application Server dimension member.
m_level	The unique code for an Application Server dimension level, as defined in the corresponding Dimension Levels table for the dimension.
d_datatype	The datatype of the dimension column in the fact tables. D_DATATYPE can be any one of the strings TINYINT, SMALLINT, INTEGER (the default), NUMBER(n), VARCHAR(n), CHAR(n). If VARCHAR or CHAR is used, then the m_code is used in character format - this option is provided since a CHAR(1) column will occupy less space than a NUMBER column in Oracle.
d_usenames	If D_USENAMES is any non 0 value, then all the fact table will have member names instead of member codes for the corresponding dimension. (the default is 0). If D_USENAMES is set, then the value in D_DATATYPE is ignored. The datatype used in the fact tables will always be VARCHAR(24) to correspond to allowed names in Application Server.
d_alias	If D_ALIAS has a non null value that string is used as the fact column name instead of m_dimname in the fact tables.
p_code	The m_code of the parent of the Application Server dimension member.
p_level	The m_level of the parent of the Application Server dimension member.
m_name	The short name for the Application Server dimension member.
p1_code	The m_code of the parent of the Application Server dimension member.
p2_code	The m_code of the grandparent of the Application Server dimension member.
pn_code	The m_code for each successive level of the Application Server dimension member. A dimension can have up to 20 levels; the maximum value for n is 18.
m_label	The long name for the Application Server dimension member.
u_label	The uppercase version of m_label.

Remarks

The *m_code* is used in the Fact tables to find the time series for a given variable and dimension member combination.

You can use the *p_code* and *p_level* columns to define the hierarchy within the dimension. If required *p3_code* and *p4_code* columns can be defined to describe more complex hierarchies. For unstructured attributes with no hierarchy, you can use a simpler version of the Dimension table which excludes the hierarchy data stored in the *p_code* and *p_level* columns.

Dimension calculations (such as the subtraction typically found in Type dimensions) are not supported in the schema. Any dimension to be exported to a schema must be usable in the

Application Server Rollup editor. Additionally, multiple hierarchies are not supported in the Dimension tables.

Note: In Hybrid OLAP models, dimension names should be of no more than 17 characters.

9.8 Dimension Delta Tables

Use

Dimension Delta tables contain changes made to Dimension tables.

Format

The Dimension Delta tables are named *prefix_dimension_D*, where *prefix* is the name of the schema prefix, and *dimension* is the name of an Application Server dimension. Each Dimension table consists of the following columns:

Column	Description
<i>change_date</i>	The date and time that the change was made.
<i>change_type</i>	The kind of change that occurred. Possible values are I (Input), U (Update), or D (Delete).
<i>m_code</i>	The unique code for an Application Server dimension member.
<i>m_level</i>	The unique code for an Application Server dimension level, as defined in the corresponding Dimension Levels table for the dimension.
<i>p_code</i>	The <i>m_code</i> of the parent of the Application Server dimension member.
<i>p_level</i>	The <i>m_level</i> of the parent of the Application Server dimension member.
<i>m_name</i>	The short name for the Application Server dimension member.
<i>p1_code</i>	The <i>m_code</i> of the grandparent of the Application Server dimension member.
<i>p2_code</i>	The <i>m_code</i> of the great-grandparent of the Application Server dimension member.
<i>m_label</i>	The long name for the Application Server dimension member.

Remarks

There is one Dimension Delta table for each Dimension table in the schema. Dimension Delta tables are not populated until the dimension is changed.

9.9 By Dimensions Table

Use

The By Dimensions table stores a unique code for combinations of Application Server dimensions. The purpose of this code is to save storage in other tables.

Format

The By Dimensions table is named *prefix_BY_DIM*, where *prefix* is the name of the schema prefix. The By Dimensions table consists of the following columns:

Column	Description
<i>by_dim_code</i>	The unique code for a specific combination of Application Server dimensions.
<i>dim_list</i>	The description for a specific combination of Application Server dimensions.

Remarks

The contents of the *by_dim_code* column are normally generated by the RDBMS sequence generator. For example, several variables in the DEMO database are dimensioned by Product, Region, and Type, while several others are dimensioned by Product and Type.

Periodicities Table

The *by_dim_code* column is a foreign key in the Variables table. That enables the dimensions upon which each variable depends to be stored as a single value in the Variables table, rather than as a list of dimensions for each variable.

The description of dimension combinations stored in the *dim_list* column is limited to 256 characters.

9.10 Periodicities Table

Use

The Periodicities table defines the supported Application Server periodicities (such as Monthly). Each periodicity is identified by a unique one-character code.

Format

The Periodicities table is named *prefix_PERIODICITIES*, where *prefix* is the name of the schema prefix. The Periodicities table consists of the following columns:

Column	Description
<i>pdv_code</i>	The unique code for a specific Application Server periodicity.
<i>pdv_name</i>	The description for a specific Application Server periodicity.

Remarks

Other tables (such as the Variables table) use the *pdv_code* column of the Periodicities table to reduce storage where periodicity is required. In such cases, *pdv_code* becomes a foreign key constraint in those tables.

Note: The HOURLY periodicity is no longer supported.

9.11 Master Table

Use

The Master table contains information required to fully model an Application Server dimensional model in the RDBMS. It contains version and fiscal calendar setting information.

Format

The Master table is named *prefix_MASTER*, where *prefix* is the name of the schema prefix. The Master table consists of the following columns:

Column	Description
<i>mst_key</i>	Unique name for each setting stored in the Master table.
<i>mst_setting</i>	The value of the setting.

Remarks

When you issue the EXPORT SCHEMA, EXPORT BASE, or EXPORT SCHEMA DATA commands, the text defining the fiscal calendar is automatically written to the Master table. Fiscal calendar information may span several rows, in which case the names stored in the *mst_key* column will be FISCAL, FISCAL1, FISCAL2...FISCAL*n*.

When you issue the IMPORT SCHEMA command, fiscal calendar information will only be updated if no variable data already exists in the Application Server model.

9.12 Month Names Table

Use

The Month Names table stores month names used in generated column names or when storing time down in fact tables and the EXPORT BASE or EXPORT SCHEMA TIME DOWN *dateformat* command uses a month name in it. This allows you to use different locales and languages in Application Server while keeping information about the month.

Format

The Month Names table is named *prefix_MONTH_NAMES*, where *prefix* is the name of the schema prefix. The Month Names table consists of the following columns:

Column	Description
<i>month_code</i>	A number between 1 and 12 that represents the month. (1=January, 2=February, and so on.)
<i>month_name</i>	A text string. For example, Jan for January.

Remark

When you issue an EXPORT command, this table is filled with default values using the locale in effect at the time the EXPORT command is run.

9.13 Day Names Table

Use

The Day Names table stores day-of-week names used in generated column names when storing time down in fact tables and the EXPORT BASE or EXPORT SCHEMA TIME DOWN *dateformat* command uses the Day name in it. This allows you to use different locales and languages in Application Server while keeping information about the day.

Note: It is not recommended to use Day names when storing time information down the column in fact tables.

Format

The Day Names table is named *prefix_DAY_NAMES*, where *prefix* is the name of the schema prefix. The Day Names table consists of the following columns:

Column	Description
<i>day_code</i>	A number between 1 and 7 that represents the day of week. (1=Sunday, 2=Monday, and so on.)
<i>day_name</i>	A text string for the day of week. For example, SUN for Sunday.

Remark

When you issue an EXPORT command, this table is filled with default values using the locale in effect at the time the EXPORT command is run.

9.14 Format Table

Use

The Format table defines a unique code to an Application Server format specification. The purpose of this code is to save storage in other tables.

Period Format Table

Format

The Format table is named *prefix_FMT*, where *prefix* is the name of the schema prefix. The Format table consists of the following columns:

Column	Description
<i>fmt_code</i>	The unique code for a specific Application Server format specification.
<i>fmt_string</i>	The description for a specific Application Server format specification.

Remarks

The *fmt_code* column is a foreign key in the Variables table. This common format enables you to store specifications as a single value in the Variables table, rather than as a repeating list for each variable.

The contents of the *fmt_code* column would usually be generated by the RDBMS sequence generator.

9.15 Period Format Table

Use

The Period Format table contains masks for each periodicity for use in generating the equivalent of P1,...Pn names in fact tables.

Format

The Period Format table is named *prefix_PERIOD_FORMAT*, where *prefix* is the name of the schema prefix. The Period Format table consists of the following columns:

Column	Description
PDY_CODE	Foreign key into the <i>prefix_PERIODICITIES</i> (PDY_CODE).
PDY_FORMAT	The mask to use in generating the column names. Formats can be myy, ymd, yyym, dmyy, dmy, mdy, ydm, yydm, yym, my, or ym, NUMERIC, or TEXT. Month can be used in place of m to specify a Month name.
START_PERIOD	Textual date as a base date. If supplied for literal strings in PDY_FORMAT, and if 2 years of monthly data is split as one year in one table and the next year in a different table and a literal MONTHNUMBER is supplied, then instead of both tables having column names MONTHNUMBER1...MONTHNUMBER12, the first year will have MONTHNUMBER1...MONTHNUMBER12, and the second will have MONTHNUMBER13...MONTHNUMBER24.

9.16 Variables Table

Use

The Variables table stores information about variables in the Application Server database.

Format

The Variables table is named *prefix_VARIABLES*, where *prefix* is the name of the schema prefix. The Variables table consists of the following columns:

Column	Description
<i>v_code</i>	The unique code for a specific Application Server variable.
<i>v_name</i>	The short name for the Application Server variable.
<i>v_alias</i>	The alias name for the Application Server variable.

<i>V_datatype</i>	Contains one of these values: TINYINT, SMALLINT, INTEGER, NUMBER(n), REAL, FLOAT, NUMBER(n,m). This column is used to determine the datatype of each variable in fact tables. For example, if V_DATATYPE is REAL for SALES, then fact tables containing SALES data will have period columns of type REAL. When you do an EXPORT VARIABLES command, the datatype is derived from the Application Server type, eg if SALES is INTEGRAL BYTES 2 in Application Server, then EXPORT will put SMALLINT in V_DATATYPE, but if SALES in Application Server is NU4 EXPORT will put REAL in V_DATATYPE. You can override this by explicitly putting a datatype on the EXPORT command line (see below).
<i>pdv_code</i>	The unique code for a specific Application Server periodicity, as defined in the Periodicities table.
<i>start_period</i>	The start period of the earliest observation for the Application Server variable.
<i>nobs</i>	The number of observations stored.
<i>tc_code</i>	The Application Server time conversion code, abbreviated to the first character. Permitted values are F (First), L (Last), A (Average), S (Sum), and W (Weighted).
<i>by_dim_code</i>	The unique code for a specific combination of Application Server dimensions, as defined in the By Dimensions table.
<i>fmt_code</i>	The unique code for a specific Application Server format specification, as defined in the Format table. This table is optional.
<i>v_label</i>	The long name for the Application Server variable.
<i>rate</i>	Boolean flag corresponding to the Application Server Rate variable characteristic.
<i>units</i>	Boolean flag corresponding to the Application Server Units variable characteristic.
<i>expense</i>	Boolean flag corresponding to the Application Server Expense variable characteristic.
<i>v_w_code</i>	The <i>v_code</i> of the variable by which the variable is weighted. This column is only applicable when <i>tc_code</i> specifies the W (Weighted) Application Server time conversion code.
<i>logic</i>	The logic field is populated for Application Server virtual variables. It contains the logic to calculate the virtual variable.
<i>integral</i>	Can have a value of 0 or 1. Corresponds to the INTEGRAL option in the Application Server CREATE VARIABLE command. Indicates that the data is stored as integers instead of double precision, so that 1, 2, or 4 bytes per data value are used instead of 8.
<i>sparse</i>	Can have a value of 0 or 1. Corresponds to the SPARSE option in the Application Server CREATE VARIABLE command. Indicates the variable contains many consecutive, identical values for the time series.
<i>bytes</i>	Can have a value of 1, 2, 4, or 8. Corresponds to the BYTES option in the Application Server CREATE VARIABLE command. Specifies the number of bytes to use per data value. The default number of bytes per data value is 8 (double precision), which allows 15 digits of precision.
<i>v_datatype</i>	This field is not currently used. Can have a value of TINYINT, SMALLINT, INTEGER, NUMBER (p,s), REAL, or FLOAT, corresponding to the RDBMS datatypes that can be used to store data in Fact tables.
Note: Currently, all data columns in Fact tables created by the EXPORT command have a datatype of FLOAT.	
<i>v_description</i>	Defines the descriptions to associate with the variables, where description is the text that describes the variable. Empty single quotation marks (' ') remove any previously set description. Corresponds to the Application Server SET VARIABLE DESCRIPTION command. Application Server displays the descriptions in the output from a SHOW VARIABLE command.

Remarks

The *v_code* for the variable will normally be generated by the RDBMS sequence generator. It is used throughout the Tracker table and Fact table to locate time series data for a variable.

Dimension Levels Table

The format of *start_period* can be in any date format accepted by Application Server. The combination of *start_period* and *nobs* is what you see in Application Server for the number of observations and date range when you issue the SHOW VARIABLE command.

9.17 Dimension Levels Table

Use

The Dimension Levels tables define the levels in Application Server dimensions. There is a Dimension Levels table for each Application Server structural dimension.

Format

The Dimension Levels tables are named *prefix_LEVELS_dimension*, where *prefix* is the name of the schema prefix, and *dimension* is the name of an Application Server dimension. Each Dimension Levels table consists of the following columns:

Column	Description
<i>m_level</i>	The unique code for an Application Server dimension level.
<i>l_name</i>	The description for an Application Server dimension level.
<i>n_members</i>	The number of members in the level.

Remarks

The value of *m_level* is used for integrity checking in the Dimension tables, Tracker table, and Fact tables.

Note: In Hybrid OLAP models, dimension names should be of no more than 17 characters.

9.18 Tracker Table

Use

The Tracker table keeps track of where data is stored and how it is to be consolidated, and as such is the most important table in the schema. The Partitions node of the Schemas folder in the Application Server Explorer tab provides you with a view directly onto the Tracker table.

Format

The Tracker table is named *prefix_TRACKER*, where *prefix* is the name of the schema prefix. The Tracker table consists of the following columns:

Column	Description
<i>v_code</i>	The unique code for a specific Application Server variable, as defined in the Variables table.
<i>l_dim</i>	The level in the dimension, where <i>dim</i> is the name of an Application Server dimension. The number of these columns corresponds to how many structural dimensions there are in the Application Server database. The values in this column relate to the dimension levels defined in the Dimension Levels table.
<i>cons_type</i>	The unique code for a specific consolidation type, as defined in the Consolidations table.
<i>nhits</i>	The number of times that this variable and level combination has been accessed. The value is updated automatically by Application Server.
<i>nrows</i>	The number of actual time series present for this variable and level combination. The value is updated automatically by Application Server.
<i>pending</i>	A Boolean flag indicating whether a Tracker table row has been marked for consolidation by the database administrator in the Partitions view. The Partitions view appears in the Application Server List tab when you click on the Partitions node of the schema you are using.

<i>cons_type_new</i>	The new consolidation type code for a specific row in the Tracker table when this is changed by the database administrator in the Partitions view. The Partitions view appears in the Application Server List tab when you click on the Partitions node of the schema you are using. The values in this column relate to the consolidation types defined in the Consolidations table.
<i>last_update</i>	The date and time on which data for this variable was last updated. This value is used to determine when data needs to be updated.
<i>min_seconds</i>	The minimum access time (in seconds) taken to retrieve this data.
<i>max_seconds</i>	The maximum access time (in seconds) taken to retrieve this data.
<i>total_seconds</i>	The total access time (in seconds) taken to retrieve this data.
<i>start_period</i>	The start period of the first column of data in the Fact table in which data is stored for this Application Server variable. If the data for this variable is stored in multiple Fact tables, additional <i>start_period</i> columns will be created in the Tracker table.
<i>nobs</i>	The number of observations stored for this variable in the specified Fact table. If the data for this variable is stored in multiple Fact tables, additional <i>nobs</i> columns will be created in the Tracker table.
<i>t_name</i>	The name of the Fact table in which data is stored for this Application Server variable. If the data for this variable is stored in multiple Fact tables, additional <i>t_name</i> columns will be created in the Tracker table.

Remarks

The Tracker table is normally created and populated when you export a schema. If the Tracker table will be created manually by a database administrator, this table must be created before any access to the schema is possible.

The data in the *nhits* column can help you decide where data is best stored, and how it is best consolidated. If data is being consolidated dynamically and the database administrator notices that the data is being accessed frequently, system performance could be improved by preconsolidating the data. Conversely, if preconsolidated data has a low hit count, it probably does not need to be preconsolidated.

The *nrows* column counts the number of actual time series present for this variable/level combination. It is used in dynamic consolidation to determine the best point from which to begin aggregation.

The *start_period*, *nobs*, and *t_name* columns form a set of columns that can be repeated if the specified time series data cannot be stored in a single Fact table. If this is the case, three new columns named *start_period1*, *nobs1*, and *t_name1* are created to store the start date, number of observations, and table name of the second Fact table in which data is stored. Further columns named *start_period2*, *nobs2*, and *t_name2* may need to be created if the data spills over to a third Fact table, and so on.

9.19 Reference Tables

Use

The Reference tables store descriptions for dimension members and variables, and provide corresponding functionality to the label feature in Application Server. You can define any number of labels for a given dimension member, enabling easy support for multi-lingual labels.

Format

The Reference tables are named *prefix_REF_dimension*, where *prefix* is the name of the schema prefix, and *dimension* is the name of an Application Server dimension. Each Reference table consists of the following columns:

Attribute Tables

Column	Description
<i>m_code</i>	The unique code for an Application Server dimension member. The <i>m_code</i> column corresponds to the <i>m_code</i> for each member in the corresponding Dimension table.
<i>language</i>	The user-defined language identifier (string). It should always be upper-case. By default Hybrid OLAP will look for EN (English).
<i>label1</i>	The description for the label. Additional columns define multiple labels for a single dimension member. Such columns are named <i>label2</i> , <i>label3</i> , and so on.

Remarks

Reference tables are optional, and are not automatically generated. They must be created by the user or the RDBMS database administrator.

You can define up to 248 alternative labels for a Hybrid OLAP dimension for each language you want to include.

Specify the *v_code* in place of the *m_code* if you are defining labels for a variable.

You can combine the unique member code (the *m_code* or *v_code*) with a *language* key to implement multiple language dependent labels.

9.20 Attribute Tables

Use

The Attribute table defines the relationships from a dimension to its attributes. For example, you could use the Attributes table to define which members of the Product dimension belonged to which flavors.

Format

The Attribute tables are named *prefix_ATT_attribute*, where *prefix* is the name of the schema prefix, and *attribute* is the name of an Application Server attribute. Each Attribute table consists of the following columns:

Column	Description
<i>m_code</i>	The unique code for an Application Server dimension member.
<i>a_code</i>	The unique code for an Application Server attribute.

Remarks

The *a_code* specified for attributes in the Attribute table corresponds to the *m_code* specified for the attribute in the corresponding Dimension table.

9.21 Table Parameters Table Reference

9.21.1 Table Parameters Table (Oracle)

Use

The (Oracle) Table Parameters table contains columns that correspond to all of the relational database table creation parameters that can be supplied as part of an Oracle SQL CREATE TABLE or CREATE INDEX statement. By amending these parameters, you can control how and where Hybrid OLAP creates the rest of the schema tables and indexes. Additionally, you can set the size for each table and index, and specify privileges for other database users on the objects you create. For Oracle8, you can specify the partitioning of tables and indexes.

Format

Table Parameter tables are named *prefix*_TABLE_PARAMS, where *prefix* is the name of the schema prefix. Each Table Parameters table for Oracle consists of the following columns; when the values are NULL, it takes the default setting that applies to the default tablespace or system-wide setting:

Column	Description
TABLENAME VARCHAR(32) NOT NULL PRIMARY KEY	<p>The name of the table to which these parameters apply. This is the primary key. The <i>tablename</i> consists of the simple name of the table, excluding any owner extension. For example, MKTING.PROMO_PERIODICITIES appears in the Table Parameters table as PROMO_PERIODICITIES.</p> <p>The <i>tablename</i> column may also contain the following wildcard characters:</p> <ul style="list-style-type: none"> * A wildcard for all base tables not explicitly mentioned. F* A wildcard for all Fact tables not explicitly mentioned. T* A wildcard for all temporary tables.
DATA_PCTFREE	<p>The PCTFREE value for the data segment.</p> <p>Note: When you expect considerable update activity in a table, specify a large PCTFREE value to reserve room for future row growth. If you do not expect a table to be updated or have rows deleted, you can specify a value of zero to conserve disk space.</p>
DATA_PCTUSED	The PCTUSED value for the data segment. The default is 40%.
DATA_INITRANS	<p>The INITRANS for the data segment. This specifies the initial number of transaction entries allocated within each data block allocated to the table. The value can range from 1 to 255 and defaults to 1.</p> <p>Note: In general, changing the INITRANS value from its default is not recommended.</p>
DATA_MAXTRANS	<p>The MAXTRANS for the data segment.</p> <p>Note: Changing the MAXTRANS value from its default is not recommended.</p>
DATA_TABLESPACE VARCHAR(20)	The name of the tablespace on which to place the data segment. If you omit this option, Oracle creates the table in the default tablespace of the user creating the table or index.
DATA_INITIAL	The size of the first database extent in bytes. The <i>data_initial</i> column can take a value (for example, 2046) or a string (for example, 2K or 5M).
DATA_NEXT	The size of the next database extent in bytes. The <i>data_next</i> column can take a value (for example, 2046) or a string (for example, 2K or 5M).
DATA_MINEXTENTS	The MINEXTENTS for the data segment. The default is one extent.
DATA_MAXEXTENTS	The MAXEXTENTS for the data segment. The default is 121 on both Windows NT and UNIX.
DATA_PCTINCREASE	The PCTINCREASE value for the data segment.

Table Parameters Table Reference

DATA_FREELISTS	<p>The FREELISTS value for the data segment. For objects other than tablespaces, FREELISTS specifies the number of free lists for each of the free list groups for the table, cluster, or index. The default (and minimum value) for this parameter is one, meaning that each free list group contains one free list.</p> <p>Note: The maximum value for this parameter depends on the data block size. If you specify a FREELISTS that is too large, Oracle7 returns an error message indicating the maximum value.</p>
DATA_FREELIST_GROUPS	<p>The FREELIST_GROUPS value for the data segment. For objects other than tablespaces, FREELIST_GROUPS specifies the number of groups of free lists for a table, cluster, or index. The default (and minimum value) for this parameter is one.</p> <p>Note: Use this parameter only if you are using Oracle7 with the Parallel Server option in parallel mode.</p>
DATA_OPTIMAL	<p>The OPTIMAL value for the data segment. The <i>data_optimal</i> column can take a value (for example, 2046) or a string (for example, 2K or 5M).</p>
DATA_PARALLEL_DEGREE	<p>The default degree of parallelism for the table in future queries. The <i>data_parallel_degree</i> column can take a value (for example, 5) or the string DEFAULT.</p>
DATA_PARALLEL_INSTANCES	<p>The default number of parallel instances for the table for future queries. The <i>data_parallel_instances</i> column can take a value (for example, 5) or the string DEFAULT.</p>
DATA_CACHE	<p>The CACHE setting for the data segment. An entry of NULL or 0 means NOCACHE; otherwise, CACHE is in effect.</p>
DATA_PARTITION	<p>The partition clause(s) to apply when the CREATE TABLE command is run, for Oracle8 only. The <i>data_partition</i> column can contain a text string of up to 2,000 characters.</p> <p>Note: The field is ignored for Oracle7, and if it contains a NULL entry.</p>
INDEX_INITTRANS	INITTRANS for the index.
INDEX_MAXTRANS	MAXTRANS for the index.
INDEX_TABLESPACE VARCHAR(20)	The tablespace in which to create any indexes.
INDEX_INITIAL	<p>The initial size of the table in bytes. The <i>index_initial</i> column can take a value (for example, 2046) or a string (for example, 2K or 5M).</p>
INDEX_NEXT	<p>The initial size of the table in bytes. The <i>index_next</i> column can take a value (for example, 2046) or a string (for example, 2K or 5M).</p>
INDEX_MINEXTENTS	The MINEXTENTS value for the index.
INDEX_MAXEXTENTS	<p>The MAXEXTENTS for the index. The <i>index_maxextents</i> column can take a value (for example, 121) or the string UNLIMITED.</p>
INDEX_PCTINCREASE	The PCTINCREASE value for the index.
INDEX_FREELISTS	The FREELISTS value for the index.

INDEX_FREELIST_GROUPS	The FREELIST_GROUPS value for the index.
INDEX_OPTIMAL	The OPTIMAL value for the index. The <i>index_optimal</i> column can take a value (for example, 2046), or a string (for example, 2K), or NULL.
INDEX_PCTFREE DECIMAL(3,0)	<p>Specifies what percentage of each index page to leave as free space when building the index. The first entry in a page is added without restriction. When additional entries are placed in an index page at least integer percent of free space is left on each page.</p> <p>The value of integer can range from 0 to 99. However, if a value greater than 10 is specified, only 10 percent free space will be left in non-leaf pages. The default is 10.</p>
INDEX_UNRECOVERABLE	Indicates whether to create the index as UNRECOVERABLE. Using this keyword makes index creation faster than using the RECOVERABLE option because redo log entries are not written. An entry of NULL or 0 means no; otherwise, YES.
INDEX_PARALLEL_DEGREE	The default degree of parallelism to create the index. The <i>index_parallel_degree</i> column can take a value (for example, 5) or the string DEFAULT.
INDEX_PARALLEL_INSTANCES	The default number of parallel instances for the table for future queries. The <i>index_parallel_instances</i> column can take a value (for example, 5) or the string DEFAULT.
INDEX_PARTITION	<p>The partition clause(s) to apply when the CREATE INDEX or ALTER TABLE ADD PRIMARY KEY commands are run, for Oracle8 only. The <i>index_partition</i> column can contain a text string of up to 2,000 characters.</p> <p>Note: The field is ignored for Oracle7, and if it contains a NULL entry.</p>
GRANT_ALL VARCHAR(250)	Comma-separated list of users/groups to grant ALL privileges to.
GRANT_ALTER VARCHAR(250)	Comma-separated list of users/groups to grant ALTER privileges to.
GRANT_INSERT VARCHAR(250)	A comma-separated list of users/groups to grant INSERT privileges to.
GRANT_REFERENCES VARCHAR(250)	Comma-separated list of users/groups to grant REFERENCES privileges to.
GRANT_SELECT VARCHAR(250)	Comma-separated list of users/groups to grant SELECT privileges to.
GRANT_UPDATE VARCHAR(250)	Comma-separated list of users/groups to grant UPDATE privileges to.
GRANT_ALL_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT ALL WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_ALTER_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT ALTER WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_DELETE_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT DELETE WITH GRANT OPTION. An entry of NULL or 0 means no;

Table Parameters Table Reference

	otherwise, YES. A null value or zero means no; otherwise, yes.
GRANT_INDEX_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT INDEX WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_INSERT_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT INSERT WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_REFERENCES_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT REFERENCES WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_SELECT_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT SELECT WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_UPDATE_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT UPDATE WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
NO_FOREIGN_KEYS DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to omit creating foreign keys on the table. Corresponds to the NOFOREIGNKEYS schema keyword in Application Server. An entry of NULL or 0 means no; otherwise, YES.
NO_ANALYZE	0 or 1. Default is 0. Specifies whether to issue an ANALYZE TABLE on the table after its loaded. Corresponds to the NOANALYZE schema keyword in Application Server. An entry of NULL or 0 means issue the ANALYZE TABLE; otherwise, do not.

Remarks

Whenever Hybrid OLAP creates a new table or index, it reads the Table Parameters table to see if the Database Administrator (DBA) has provided any extra information about how or where to create it. If relevant information is included in the Table Parameters table, Hybrid OLAP will use it; otherwise, wherever possible, Hybrid OLAP will estimate and create the object with a reasonable size.

If the DBA has not specified in the Table Parameters table which tablespace to put the object on, but has supplied a TABLESPACE or INDEXSPACE keyword to an EXPORT or CONSOLIDATE PENDING (Schema) command, Hybrid OLAP will use the tablespace or indexspace specified in the keyword. If no placement details are given in either the Table Parameters table or these keywords, then the object will be placed in the user's default tablespace.

For more information about the database parameters that can be set in the Table Parameters table, see the *Oracle System Administrator's Guide*.

For more information about Oracle8 partitioning, see the *Oracle8 Reference Manual*.

Notes:

Because the Table Parameters table can control the creation of the other base tables in the schema, it has to be created prior to, and independent of, them. This is done through the SCHEMA subsystem EXPORT MASTER command. Issue the EXPORT MASTER command before issuing the EXPORT BASE or EXPORT SCHEMA commands if you want to maintain this level of control. If the EXPORT BASE or EXPORT SCHEMA command is issued without a prior EXPORT MASTER command, the Table Parameters table will still be created, and will be populated with as many reasonable values as possible. However, explicit Database Administrator control over many elements of the object created during the schema export process is lost.

Schemas created in Hybrid OLAP versions prior to 6.1 do not contain a Table Parameters table. These schemas may still be used; however, you cannot control table placement, size, or privileges when new tables and indexes are created.

The format of the Table Parameters table differs significantly for Oracle and DB2.

9.21.2 An Overview: Partitioning in Oracle8

The basic idea of partitioning in Oracle8 is to divide a large table into several smaller tables, or subtables, while still allowing the user to access the whole table as a single object. Oracle manages the subdivision of the table into smaller parts transparently. The same partitioning can be applied to indexes.

Partitioning has several advantages:

The performance against several smaller tables is likely to be faster than the performance against one large table:

- Each subtable has fewer rows, and can therefore be scanned faster.
- Each subindex will be smaller, and so will be faster to build and query.
- A query may only involve access to one subtable instead of the whole table; if it involves access to several subqueries, Oracle may be able to parallelize it.
- Each subtable and subindex can be placed on different disks to improve performance.

The sub tables are easier to manage; each can be loaded separately or truncated separately, and possibly in parallel.

It is more secure since each subtable can be backed up and recovered separately, thus minimizing the risk of losing all the data due to a system failure.

Specifying a partition clause

You can tell Oracle how to split a table into subtables by adding a partition clause to the end of the CREATE TABLE, CREATE INDEX, or ALTER TABLE ADD PRIMARY KEY commands.

For example, you could divide a large table containing names and addresses into four subtables alphabetically on names, as follows:

```
create table mytable (name varchar2(32),address varchar2(64))
partition by range(name) (
partition part1 values less than ('F') tablespace Kathy pctfree 10 storage(initial 100k nect 100k),
partition part2 values less than ('N') ,
partition part3 values less than ('T') tablespace Jay,
partition part4 values less than (MAXVALUE) tablespace Peter)
```

The user accesses the table as *mytable*, but internally there are four subtables. Note that the partition statement:

```
partition by range(name) (
partition part1 values less than ('F') tablespace Kathy pctfree 10 storage(initial 100k nect 100k),
partition part2 values less than ('N') ,
partition part3 values less than ('T') tablespace Jay,
partition part4 values less than (MAXVALUE) tablespace Peter)
```

is held in the *data_partition* column in the Table Parameters table in Hybrid OLAP.

Table Parameters Table Reference

If required, the subtables can be accessed and maintained individually via SQL. The following example queries just *subtable part3* and returns names beginning with N through S:

```
select * from mytable partition (part3)
```

Note that you can apply tablespace and storage details to each subtable separately. The following example creates a partitioned index broken down by the same criteria as the parent table (indicated by the *local* keyword) and specifies storage details for each subindex:

```
create index myindex on mytable(name)
```

```
local (
```

```
partition part1 tablespace tom storage (initial 20k),
```

```
partition part4 storage (minextents 10))
```

The subindex details are optional, so the partition clause could simply be just *local*. Note that the partition statement:

```
local (
```

```
partition part1 tablespace tom storage (initial 20k),
```

```
partition part4 storage (minextents 10))
```

is held in the *index_partition* column in the Table Parameters table in Hybrid OLAP.

Where you want subdivision by more than one column, you put a list of column names in the *range* clause, and range values in the *values less than* clause in the same order. In Hybrid OLAP, the obvious way to use this is to partition by V_CODE in the Tracker and Fact tables, since V_CODE forms the leading part of the primary key in both; in Dimension tables, M_CODE is a suitable candidate.

9.21.3 Example: Partitioning in Oracle8

...This example shows how the user might implement partitioning in

...Hybrid OLAP using the Juice database as an example. Note that for

...clarity, this example does not include tablespace and storage

...clauses; obviously, in a real implementation, the Database

...Administrator would certainly include those as well.

prefix XXX

EXPORT BASE

...modify the Table Parameters table for large dimensions

```
sql update xxx_table_params set data_partition = 'partition by range(v_code) (partition part1 values less than (11),partition part2 values less than (21),
```

```
partition part3 values less than (31), partition part4 values less than (MAXVALUE))' where tablename = 'XXX_PRODUCT'
```

```
sql update xxx_table_params set index_partition = 'local' where tablename='XXX_PRODUCT'
```

```
sql commit
```

EXP DIM *

EXP VAR *

...modify the Table Parameters table for the Tracker table

```
sql update xxx_table_params set data_partition = 'partition by range(v_code) (partition part1 values less than (3),partition part2 values less than (9),
```

```
partition part3 values less than (12), partition part4 values less than (MAXVALUE))' where tablename = 'XXX_TRACKER'
```

```

sql update xxx_table_params set index_partition = 'local' where tablename='XXX_TRACKER'
sql commit
TRACKER INS *
...modify the Table Parameters table for the Fact tables
sql update xxx_table_params set data_partition = 'partition by range(v_code) (partition part1 values
less than (3),partition part2 values less than (9),
partition part3 values less than (12), partition part4 values less than (MAXVALUE))' where tablename =
'XXX_FACT_3_18'
sql update xxx_table_params set index_partition = 'local' where tablename='XXX_XXX_FACT_3_18'
sql commit
EXP DATA *

```

9.21.4 Table parameters Table (DB2)

Use

The (DB2) Table Parameters table contains columns that control the properties, placement, and GRANT privileges to be used whenever you create a Hybrid OLAP table.

By editing these parameters, you can control how and where Hybrid OLAP creates the rest of the schema tables and indexes. Additionally, you can set the size for each table and index, and specify privileges for other database users on the objects you create.

Format

Table Parameter tables are named *prefix_TABLE_PARAMS*, where *prefix* is the name of the schema prefix. Each Table Parameters table for DB2 consists of the following columns; when the values are NULL, it takes the default setting that applies to the default tablespace or system-wide setting:

Column	Description
TABlename VARCHAR(32) NOT NULL PRIMARY KEY	<p>The name of the table to which these parameters apply. This is the primary key. The <i>tablename</i> consists of the simple name of the table, excluding any owner extension. For example, MKTING.PROMO_PERIODICITIES appears in the Table Parameters table as PROMO_PERIODICITIES.</p> <p>The <i>tablename</i> column may also contain the following wildcard characters:</p> <ul style="list-style-type: none"> * A wildcard for all base tables not explicitly mentioned. F* A wildcard for all Fact tables not explicitly mentioned. T* A wildcard for all temporary tables.
DATA_TABLESPACE VARCHAR(20)	The tablespace in which you create the table.
DATA_PCTFREE DECIMAL(3,0)	<p>Specifies the percentage of each data page to leave as free space when inserting rows into the table. The value of integer can range from 0 to 99. The default is 0.</p> <p>This table parameter is only used when using the LOAD or REORGANIZE TABLE utilities in DB2. Standard methods of inserting rows into tables ignore this setting completely and fill as many rows into each data block as possible. Hybrid OLAP can use the LOAD facility via a DB2 API when the</p>

Table Parameters Table Reference

	DIRECTLOAD keyword is used in EXPORT and the Hybrid OLAP transformer. REORGANIZE TABLE can only be run from within the DB2 CLP environment.
INDEX_CLUSTER DECIMAL(1,0)	0 or 1 – whether to create the primary key index as CLUSTERED or NON CLUSTERED (the default is 0). The cluster factor of a clustering index is maintained or improved dynamically as data is inserted into the associated table, by attempting to insert new rows physically close to the rows for which the key values of this index are in the same range. Only one clustering index may exist for a table so CLUSTER may not be specified if it was used in the definition of any existing index on the table (SQLSTATE 55012). A clustering index may not be created on a table that is defined to use append mode (SQLSTATE 428D8). This is likely to be expensive on large tables which have frequent INSERT/DELETE operation.
INDEX_TABLESPACE VARCHAR(20)	The tablespace in which to create any indexes. Note: DB2 only allows you to place indexes for a table on a different tablespace than the tablespace where the table data resides if you are using DMS (Database managed Tablespaces). If you are using SMS (System Managed Tablespaces), DB2 mandates that the index must be on the same tablespace as the table data. If you specify INDEX_SPACE to be different than DATA_TABLESPACE, you will receive DB2 errors when Hybrid OLAP tries to create the index. Also DB2 only allows the specification of an index tablespace with an explicit data tablespace. Hybrid OLAP code will ignore any setting in INDEX_TABLESPACE if DATA_TABLESPACE is NULL.
INDEX_PCTFREE DECIMAL(3,0)	Specifies what percentage of each index page to leave as free space when building the index. The first entry in a page is added without restriction. When additional entries are placed in an index page at least integer percent of free space is left on each page. The value of integer can range from 0 to 99. However, if a value greater than 10 is specified, only 10 percent free space will be left in non-leaf pages. The default is 10.
INDEX_MINPCTUSED DECIMAL(3,0)	Indicates whether indexes are reorganized online and the threshold for the minimum percentage of space used on an index leaf page. If after a key is deleted from an index leaf page, the percentage of space used on the page is at or below integer percentage, an attempt is made to merge the remaining keys on this page with those of a neighboring page. If there is sufficient space on one of these pages, the merge is performed and one of the pages is deleted. The value of integer can be from 0 to 99. However, a value of 50 or below is recommended for performance reasons.
INDEX_ALLOWREVERSESCANS DECIMAL(1,0)	Specifies that an index can support both forward and reverse scans; that is, in the order defined at INDEX CREATE time and in the opposite (or reverse) order. Value is 0 or 1. The default is 0.
GRANT_ALL VARCHAR(250)	Comma-separated list of users/groups to grant ALL privileges to.
GRANT_ALTER	Comma-separated list of users/groups to grant ALTER

VARCHAR(250)	privileges to.
GRANT_CONTROL VARCHAR(250)	Comma-separated list of users/groups to grant CONTROL privileges to.
GRANT_DELETE VARCHAR(250)	Comma-separated list of users/groups to grant DELETE privileges to.
GRANT_INDEX VARCHAR(250)	Comma-separated list of users/groups to grant INDEX privileges to.
GRANT_INSERT VARCHAR(250)	Comma-separated list of users/groups to grant INSERT privileges to.
GRANT_REFERENCES VARCHAR(250)	Comma-separated list of users/groups to grant REFERENCES privileges to.
GRANT_SELECT VARCHAR(250)	Comma-separated list of users/groups to grant SELECT privileges to.
GRANT_UPDATE VARCHAR(250)	Comma-separated list of users/groups to grant UPDATE privileges to.
GRANT_ALL_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT ALL WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_ALTER_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT ALTER WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_CONTROL_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT CONTROL WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_DELETE_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT DELETE WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_INDEX_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT INDEX WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_INSERT_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT INSERT WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_REFERENCES_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT REFERENCES WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_SELECT_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT SELECT WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
GRANT_UPDATE_WGO DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to give GRANT UPDATE WITH GRANT OPTION. An entry of NULL or 0 means no; otherwise, YES.
NO_FOREIGN_KEYS DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to omit creating foreign keys on the table. An entry of NULL or 0 means no; otherwise, YES.
NO_INITIAL_LOGGING DECIMAL(1,0)	0 or 1. Default is 0. Specifies whether to create the table with NOT LOGGED INITIALLY. An entry of NULL or 0 means no;

otherwise, YES.

Any changes made to the table by an Insert, Delete, Update, Create Index, Drop Index, or Alter Table operation in the same unit of work in which the table is created are not logged. All catalog changes and storage-related information are logged, as are all operations that are done on the table in subsequent units of work.

A foreign key constraint cannot be defined on a table that references a parent with the NOT LOGGED INITIALLY attribute. This can be used to improve performance on loading the table with conventional inserts etc and is similar to the UNRECOVERABLE option we use in Oracle. This applies only to the table and not the index.

Foreign keys are not allowed on such a table as a parent table since the contents are not logged and therefore unsafe – if the database were rolled back, the foreign key integrity constraint could not be maintained. Therefore this option must be ignored on most of our tables if foreign keys are created. It is most useful for fact tables.

NO_RUNSTATS
DECIMAL(1,0)

0 or 1. Default is 0. Specifies whether to omit doing a RUNSTATS on a table after it is loaded with rows. The RUNSTATS command is a CLP command that gathers the statistics required by the Query Optimizer to decide how best to execute a query.

Remarks

Whenever Hybrid OLAP creates a new table or index, it reads the Table Parameters table to see if the Database Administrator (DBA) has provided any extra information about how or where to create it. If relevant information is included in the Table Parameters table, Hybrid OLAP will use it; otherwise, wherever possible, Hybrid OLAP will estimate and create the object with a reasonable size.

If the DBA has not specified in the Table Parameters table which tablespace to put the object on, but has supplied a TABLESPACE or INDEXSPACE keyword to an EXPORT or CONSOLIDATE PENDING (Schema) command, Hybrid OLAP will use the tablespace or indexspace specified in the keyword. If no placement details are given in either the Table Parameters table or these keywords, then the object will be placed in the user's default tablespace.

Notes:

Because the Table Parameters table can control the creation of the other base tables in the schema, it has to be created prior to, and independent of, them. This is done through the SCHEMA subsystem EXPORT MASTER command. Issue the EXPORT MASTER command before issuing the EXPORT BASE or EXPORT SCHEMA commands if you want to maintain this level of control. If the EXPORT BASE or EXPORT SCHEMA command is issued without a prior EXPORT MASTER command, the Table Parameters table will still be created, and will be populated with as many reasonable values as possible. However, explicit Database Administrator control over many elements of the object created during the schema export process is lost.

Schemas created in Hybrid OLAP versions prior to 6.1 do not contain a Table Parameters table. These schemas may still be used; however, you cannot control table placement, size, or privileges when new tables and indexes are created.

The format of the Table Parameters table differs significantly for Oracle and DB2.

10 Hybrid OLAP Reference Topics

10.1 Extensions and Constraints to Application Server

The Application Server features described in this section behave differently in Hybrid OLAP models. In some cases, Hybrid OLAP may limit or replace existing functionality; in others, functionality has been expanded to take advantage of the Hybrid OLAP enhancements to Application Server.

ALLOCATE command constraint

In Hybrid OLAP models, always use the UPDATE keyword with the Application Server ALLOCATE command.

CALCULATE command constraint

In a Hybrid OLAP model that has already been exported, a `CALCULATE variable = expression` statement (such as `CALCULATE Sales=4`) works as if it were a standard Application Server model with the variable set to NOCONSOLIDATE, regardless of whether the variable in the Hybrid OLAP is set to NOCONSOLIDATE.

CONSOLIDATE command constraints

Pure summation is the only kind of consolidation supported on variables that are dimensioned by a Hybrid OLAP dimension.

In Hybrid OLAP models, CONSOLIDATE PENDING is used. Regular Application Server CONSOLIDATE will not work in a Hybrid OLAP model.

CREATE VARIABLE command constraint

You cannot create a new variable in an existing Hybrid OLAP model by using the Application Server CREATE VARIABLE LIKE keyword. Instead, create the new variable using the CREATE VARIABLE *variablename* BY... syntax, and then export the newly created variable.

CREATE VERSION command constraint

The Application Server CREATE VERSION command is not implemented in Hybrid OLAP.

Dimension name constraints

Dimension names must be in all upper-case in the RDBMS tables. When you use the EXPORT (Schema) commands, this happens automatically.

When you are building applications, any drillthrough or guest members must reference the short name in the dimension table, not the long name (the label).

EXHIBIT COUNT command extensions

In Hybrid OLAP models, the EXHIBIT COUNT command functions as follows:

EXHIBIT COUNT *dimensionname* only gives the counts stored inside Application Server. Counts in the RDBMS are not included.

EXHIBIT COUNT *dimensionname* BYLEVEL produces a line of tabbed output that displays the number of levels in the dimension, followed by a list of the counts for the members in each of those levels. It only gives the counts stored inside Application Server. Counts in the RDBMS are not included.

EXHIBIT COUNT *dimensionname* DRILLTHRU looks in the schema. The counts for the inputs, outputs, and results refer to the numbers in the schema. These counts disregard the dimension as it appears in Application Server.

EXHIBIT COUNT *dimensionname* DRILLTHRU BYLEVEL looks in the schema to give the number of external levels and the number of external members at each level. These counts disregard the dimension as it appears in Application Server.

KEY command constraint

In Hybrid OLAP models, always use the BOTH keyword with the Application Server KEY command.

ORDER command extension

DRILLTHRU indicates that the command will do the ordering using the dimension in the relational schema, and ignore the selection in Application Server. For more information, see ORDER DRILLTHRU (Schema) in the Hybrid OLAP command reference.

ROLLUP editor constraint

The Application Server ROLLUP editor, used to define which member combinations (quadrants) Application Server uses in the fast consolidation, has no effect in a Hybrid OLAP model. All information is controlled through the Tracker table.

SAVE STATUS command constraint

In Hybrid OLAP models, always use the SYMBOLIC keyword with the Application Server SAVE STATUS command. This will prevent hash number (#) selects, which are not implemented in Hybrid OLAP.

SELECT command constraints

Do not use the SELECT keyword USING in Hybrid OLAP models.

Complex SELECT WHERE commands work, but may be slower than in native Application Server.

Application Server features not available in Hybrid OLAP

The following Application Server features are currently not available when you are utilizing Hybrid OLAP functionality:

Multiple hierarchies in dimensions

Dimension classes

Dimensional Editor Rollup relationships other than SUM (no negative summing)

Text variables

Hash number selections

Multiple parents

Note: Once a Hybrid model is created, you cannot change its physical filename or add the model to MASTERDB using a different name.

10.2 Maintaining Your Environment

10.2.1 Creating a Reference Table for Your Schema

In Hybrid OLAP schemas, Reference tables store descriptions for dimension members and variables, and provide corresponding functionality to the label feature in Application Server. You

can define any number of labels for a given dimension member, enabling easy support for multi-lingual labels.

Reference tables are optional, and are not automatically generated. They must be created by the user or the RDBMS database administrator.

The SQL format to create a Reference table is as follows, for a prefix MYPREFIX and a dimension PRODUCT:

```
CREATE TABLE MYPREFIX_REF_PRODUCT AS (M_CODE INTEGER PRIMARY KEY,LANGUAGE  
  CHAR VARYING(n),  
  ALTNAME1 CHAR VARYING(n1),  
  ALTNAME2 CHAR VARYING(n2),  
  .  
  .  
  ALTNAMEx CHAR VARYING(nx),  
  CONSTRAINT some_name M_CODE REFERENCES MYPREFIX_PRODUCT(M_CODE))
```

where n, n1, n2,...nx are widths that you determine. ALTNAME is a placeholder here; and may be replaced by more descriptive names.

LANGUAGE is a column that can contain user-defined language strings, such as FR or FRA or FRENCH. However, by default, Hybrid OLAP looks for EN for English. The contents of the LANGUAGE column must be in upper-case.

For example, in a Reference table for PRODUCT with columns MYNAMES1, MYNAMES2, and MYNAMES3, and languages of EN and FR, the Hybrid OLAP command

```
SET LABEL PRODUCT FROM MYLINKID PREFIX MYPREFIX NAME MYNAMES2
```

will use the column MYNAMES2 for language EN in the Reference table to get the labels to use in displays in Application Server. The Hybrid OLAP command

```
SET LABEL PRODUCT FROM MYLINKID PREFIX MYPREFIX NAME MYNAMES3 LANGUAGE FR
```

will get the labels from Reference table column MYNAMES3 where the LANGUAGE column is FR.

10.2.2 Maintaining the Schema Administration Tables

Schemas you create at the IDQL command line are not automatically maintained by the Schema Administration tables.

If you delete a schema that is listed in the Schema Administration tables from the IDQL command line using the Schema subsystem DROP commands, the corresponding records in the Schema Administration tables will not be deleted.

You can use your RDBMS's SQL tool to delete individual records, as follows:

1. In the PILOT_SCHEMA table, delete the record that describes the invalid schema. Before you delete the record, make a note of the SCHEMA_CODE value.
2. In the PILOT_SCHEMA_MODEL table, delete the record with the SCHEMA_CODE value that matches the record you deleted from the PILOT_SCHEMA table.

10.2.3 Estimating Hybrid OLAP Custom Work Database Size

When you issue the CONSOLIDATE PENDING (Schema) command, the Work database is used for working data storage and caching. The Work database size will reflect the size of the dimensions, which, in Hybrid OLAP models, can be very large.

When working storage (and thus the Work database) is likely to be large, you should specify an alternate Work database, create it as a partitioned database, and give it enough space to handle the model.

To estimate the maximum amount of space you need for your Work database, apply the following formula:

$$((\text{number of periods in current SET PERIOD}) + 2) * 8 * X * Y$$

where X is the maximum number of rows for a variable/level combination.

For example, suppose that you have most existing data for SALES and you have 3 dimensions A, B, and C, each with 3 levels with (1000,10,1), (2000,20,1), and (3000,30,1) members at each level respectively. You estimate that approximately 10% of all possible combinations exist.

In this case, the value of X could be as large as

$$((1000*20*30) + (2000*10*30) + (3000*10*20)) * 0.1$$

or 180K.

Y is a safety factor with a suggested value of 2. This takes into account internal rollups of consolidated combinations into multiple parent combinations.

In the example here, if we have 24 periods, then the Work database could grow to approximately 75MB. You should take the estimate created with this formula into account when you create the alternate Work database.

10.2.4 Tracking the Successful Completion of CONSOLIDATE PENDING

While executing the CONSOLIDATE PENDING command, Hybrid OLAP does regular internal commits. If anything goes wrong during the consolidation, you cannot do a complete rollback to the start of the process. Additionally, if you run out of rollback space or tablespace, you may be left with a partially complete consolidation.

You can use the Tracker table to troubleshoot incomplete consolidations. Sections you had marked as pending consolidation that completed successfully will have the Tracker table PENDING column reset to 0, and a LAST_UPDATE time later than the time at which you started the consolidation. Sections you had marked as pending that did not complete successfully will have a LAST_UPDATE time earlier than the start of the consolidation. In the failed case, the PENDING column will not be reset to 0.

If you discover that your consolidation was not complete, reissue the CONSOLIDATE PENDING command, but only for those sections that failed.

Note: Use of the NORESET keyword will prevent Hybrid OLAP from resetting the contents of the Tracker table PENDING column.

10.2.5 Manually Setting Fact Table Partitioning and Placement

In Hybrid OLAP models, the actual time series data is stored in Fact tables. Data for some time series may span across several Fact tables. Fact table creation and settings are determined in the Tracker table.

By default, Hybrid OLAP automatically splits Fact table data over time. However, in many cases, greater partitioning is required – to manage the size of the Fact tables and improve the performance of loading, index creation, and subsequent retrieval. In such cases, you can manually

split Fact table data by time and variable across multiple disks, using your own criteria based on the structure of your data, your analysis needs, and your system.

For example, if you have multiple time periods and several years' worth of data, Hybrid OLAP might place all of the data into a single Fact table by default. However, if you have determined that you will need to access the last 12 months' worth of data more often, you might decide to place that data into one Fact table, and the rest into a second, "history" Fact table. The majority of subsequent queries will run faster.

To split fact data manually, you do not have to physically create any of the partition tables. After populating the Tracker table with rows for the variable and level combinations, and before any Fact tables are created, update the Tracker table to reflect your chosen partitioning scheme. To split the fact data across many smaller RDBMS tables, add additional rows using your RDBMS's SQL utility, and update the rows with different table names for different combinations.

When a CONSOLIDATE PENDING or CALCULATE takes place, the Tracker table rows are used to determine where to put the data. If a combination is marked to be stored in the RDBMS, the table names (by time and combination) that you supplied are used. Where those tables do not already exist, they are created for you, together with indexes.

10.2.6 Recommended Settings & Memory for Optimal HOLAP Performance (Oracle)

When carrying out a sort to disk, the following should be considered:

For Oracle, try to give the RDBMS more memory for sorting by adjusting the init.ora parameter SORT_AREA_SIZE, which determines the number of bytes each process can use for sorting.

It is recommended that the temporary tablespace or (tempdb) is located on a different physical disk to the tablespace or segment from the table itself, in order to reduce disk contention.

Try to create the index on a different tablespace or segment and on a different physical disk to both the temporary sort space and the table being sorted. Placing each component on a different disk reduces disk contention, takes advantage of multiple disk caches, and provides a degree of parallelism.

10.2.7 Recommended Settings and Memory for Optimal HOLAP Performance (DB2)

When carrying out a sort to disk, the following should be considered:

For DB2, memory used for sorting is governed by two system parameters:

- Sort Heap Threshold (sheapthres) which is an instance setting.
- Sort Heap Size (sortheap) which is a database specific setting.

These parameters are documented in the *DB2 Administrator's Guide*. The default value for sortheap allows 1MB for sorting. This amount should be enough for most of our queries, but the database administrator may want to monitor this and make sortheap higher if appropriate.

It is recommended that the temporary tablespace or (tempdb) is located on a different physical disk to the tablespace or segment from the table itself, in order to reduce disk contention. The default system temporary tablespace is TEMPSPACE1. To use GLOBAL TEMPORARY TABLES, the user must also create a user temporary tablespace. This should be on a separate physical disk to the other tablespaces if possible.

Try to create the index on a different tablespace or segment and on a different physical disk to both the temporary sort space and the table being sorted. Placing each component on a different disk reduces disk contention, takes advantage of multiple disk caches, and provides a degree of parallelism.

You can only do this in DB2 if you are using DB2 DATABASE MANAGED TABLESPACES (DMS tablespaces). If you place tables on SYSTEM MANAGED TABLESPACES (SMS tablespaces), then DB2 mandates that the tables and all its indexes reside in the same tablespace.

10.2.8 Using the Oracle Sql*Loader DIRECT Load Path

When loading very large numbers of rows into Oracle tables, the conventional method of inserting millions of rows into a table with a complex primary key using a SQL insert statement (even using array processing to insert many rows in each call to Oracle) is unacceptably slow. The creation of the index itself is also extremely slow, even when the index is created after the rows are inserted.

To overcome this problem of scale, the Oracle Sql*Loader DIRECT load path provides a more efficient means of loading rows into tables and creating the indexes at the same time, and is much faster than the conventional method. For extremely large implementations, where Fact tables could contain tens of millions of rows, using Sql*Loader with the DIRECT path load is the only way to insert this number of rows in an acceptable time frame. Hybrid OLAP has been enhanced so that it can invoke Sql*Loader in this way.

Constraints when using the DIRECT load path

In order to use the DIRECT load path, the following must be true:

Sql*Loader must have exclusive access to the table, and to any indexes on the table; this removes the requirement for carrying out the usual time consuming multi-user locking. The Database Administrator must either shut down Oracle and restart it in RESTRICTED mode, or disable any Oracle logins, or be sure that no other user is going to access the table that Sql*Loader is using.

To carry out a binary load, the machine on which Sql*Loader is running, and the target Oracle machine must be of the same type, that is, both NT Intel, or both Solaris, or both HP-UX – you cannot run a binary load from an NT Intel machine to a Solaris machine.

To ensure that binary or ASCII direct loads run reliably, it is suggested that the source and target machines should be running exactly the same version of the operating system, as well as being of the same machine type.

Although ASCII loads can be processed via Sql*Net, this may be slower than binary loads, thus reducing the benefits of carrying out the DIRECT load.

When to use the DIRECT load path

It would be beneficial to use the DIRECT load path when you are inserting many rows into any empty table, or loading a sizable fraction of the number of rows which already exist into a non-empty table. You would not want to use DIRECT load path if you are loading a small number of rows into a table, whether it is empty or not, but particularly when it already contains many rows.

How to use the DIRECT load path

It is not possible for HOLAP to invoke Sql*Loader and execute a DIRECT path load automatically – the user must tell HOLAP to do so. To invoke the DIRECT load path of Sql*Loader, you specify `DIRECT=TRUE` on the command line when starting Sql*Loader.

The `DIRECTLOADPATH` keyword has also been added to the `EXPORT DATA`, `EXPORT DIMENSION`, `CONSOLIDATE PENDING`, `CONSOLIDATE`, `CALCULATE`, and `TRACKER INSERT` commands. There is also a `DIRECTLOAD` setting in the Transformer – for more information, see the Transformer online Help file.

When you specify **DIRECTLOAD**, HOLAP runs Sql*Loader and invokes it with **DIRECT=TRUE**. When it creates any Dimension or Fact tables, the primary key and any other indexes are created when the tables are created, rather than at the end of the processing when all the rows have been loaded, which allows the Sql*Loader to populate the indexes faster.

The **DIRECTLOAD** keyword is ignored if HOLAP is not going to do pure inserts on a Fact table, for example, if you specify **CONSOLIDATE PENDING DIRECTLOAD SALES** where the Fact table already contains existing rows for **SALES**.

Updating Isdal.ini to use **DIRECT** load path

As well as specifying the **DIRECTLOAD** keyword, you must add the following section to your Isdal.ini file:

```
[SqlLoader]
Exe=executable
Temp=tempdir
Mode=mode
Method=method
PipeSize=size
Rows=rows
```

where:

<i>executable</i>	is the name of the Sql*Loader executable file. You must also specify the location of the file if it is not on your path.
<i>tempdir</i>	specifies the name of a temporary directory where files required by Sql*Loader can be created.
<i>mode</i>	can be either ASCII or BINARY. If you specify ASCII, the data is converted to ASCII format before being sent to the Sql*Loader. You must specify ASCII if the client and target are not exactly the same type of machine.
<i>method</i>	can be either FILES or PIPES.

As Sql*Loader is a command line program, the input data is normally expected to be in files. When **FILES** is specified, the data will be written to files in the specified *temp* directory. If the processing is successful and there are no errors, these files will be deleted automatically (unless **LEAVE=1**). If you specify **FILES**, you will require extra disk space to store those files, and processing will be slower than using a pipe.

Specify **PIPES** to use a named pipe to transfer data to Sql*Loader, which is faster and uses much less disk space than **FILES**.

<i>size</i>	is the number of bytes to use as a buffer for the named pipe. On Windows NT, a buffer of 256K should be sufficient. On UNIX, the maximum size is platform-dependent. If <i>size</i> is not specified, a default value of 32K is used.
<i>rows</i>	is the number of rows Sql*Loader will insert before carrying out an internal save (similar to a COMMIT). A value of 1,000,000 may be appropriate. At present, by default, an internal save is not carried out until all rows have been inserted into the table. For more information, refer to the Oracle documentation.

11 Transformer

11.1 What Is the Transformer?

The Transformer is a program that allows you to extend your Hybrid OLAP model by performing complex data transformation and large-volume data loads.

The Transformer converts transactional data from its source format to the required Hybrid OLAP schema table formats.

Transformer input can come from an ASCII text file or a Link ID source. Output can either populate the schema dynamically or be sent to an output file to be used with a native SQL loader.

The Transformer is a command line utility that has the same system requirements as Hybrid OLAP. It is installed on the server as part of your Application Server installation. You provide source and target data settings through a parameter file.

A Transformer parameter (initialization) file consists of named sections that refer to the dimension or fact data you are modifying. You can create a Transformer parameter file in any text editor. Rather than start from scratch, you can open, modify, and rename an existing Transformer parameter file. You can create single or multiple sections in a Transformer parameter file.

By changing the Truncate= parameter from Yes to No in the initialization file, you can perform an incremental update to load new dimension members, change dimension member relationships, and load two kinds of variable information: attribute and time series (Fact). New data can be written to blank schema tables as well.

11.2 Steps to Take Before Running the Transformer

Before running the Transformer, the data to be loaded should be made available, either in a text file or in a data source that the Transformer can reference through a Link ID. The Transformer configuration file also needs to be set up correctly, to describe how the data is to be added to your schema.

Before making any changes, it is a good idea to check the schema to make sure that the changes are valid. For example, before using the Transformer to delete stores from your retail schema, make sure that the stores exist.

11.3 Creating a Transformer Parameter File

A Transformer parameter (or initialization) file consists of named sections that refer to the dimension, attribute, or fact data you are modifying. You can create a Transformer parameter file in any text editor. Rather than start from scratch, you can open, modify, and rename an existing Transformer parameter file.

For more information about settings for the Transformer parameter file, click on the appropriate section below:

All sections

Dimension sections

Attribute sections

Fact sections

11.4 Running the Transformer

Procedure to run the Transformer from a DOS command prompt

1. The Transformer is run at a command prompt. Make sure that the Transformer executable, SGTRANS.EXE, is in the current directory or that the location is in the path statement.
2. Issue the following command:

```
sgtrans -inifile <full path to ini file> -s <section name> [-l <full path to log file> [-o]]
```

For example, to load a section called [GeogLoad] found in a parameter file called SGTRANS.INI, found in the \Program Files\Pilot Software\Common\home directory, issue the following command:

```
sgtrans -inifile\Program Files\Pilot Software\Common\home\sgtrans.ini -s GeogLoad -l \Program Files\Pilot Software\Common\home translog
```

Note: You can direct the output to a specified log file. If the specified file already exists, by default, the -l option appends the output to the file; you must additionally specify the -o option to overwrite the contents of any existing file.

Procedure to run the Transformer from a UNIX command prompt

1. The Transformer is run at a command prompt. Make sure that the Transformer file, sgtrans, is in the current directory or that the location is in the path statement.
2. Issue the following command:

```
sgtrans -inifile <full path to ini file> -s <section name> [-l <full path to log file> [-o]]
```

For example, to load a section called [GeogLoad] found in a parameter file called SGTRANS.INI, found in /Program Files/Pilot Software/Common/home directory, issue the following command:

```
sgtrans -inifile/program files/pilot software/Common/home/sgtrans.ini -s GeogLoad -l /program files/pilot software/Common/home/translog
```

Note: You can direct the output to a specified log file. If the specified file already exists, by default, the -l option appends the output to the file; you must additionally specify the -o option to overwrite the contents of any existing file.

11.5 Parameter File Reference

11.5.1 Sample Transformer Parameter File

```
; Location of Application Server message file
[windows]
TBDB=c:\Program Files\Software\Common\Data\tbdb.eng

; Dimension load with source data in relational
[customer_sql]
Loading=Dimension
Schema=lss
Table=customer
```

Parameter File Reference

LinkIDSource=orasource

SourceSQL=select CUSTOMER, CUSTOMERNA, DISTRICT, DISTRICTNA, REGION,
REGIONNA from customer

Format=Delimited TAB

LinkIDTarget=holap

Result=Yes

Truncate=Yes

Trace=No

; Match fields returned from RDBMS with levels and labels in dimension

Level1=CUSTOMER

Level2=DISTRICT

Level3=REGION

Label1=CUSTOMERNA

Label2=DISTRICTNA

Label3=REGIONNA

; Dimension load with source data in flat file

[customer_text]

Loading=Dimension

Schema=lss

Table=customer

InputFile=customer.txt

Format=Delimited,

Delimiter="

LinkIDTarget=holap

Result=Yes

; Change Truncate to No after initial load to add dimension members

Truncate=Yes

Trace=No

; Describe columns in flat file

Col1=CUSTOMER,TEXT 10

Col2=CUSTOMER_LBL,TEXT 50

Col3=DISTRICT,TEXT 10

Col4=DISTRICT_LBL,TEXT 50

Col5=REGION,TEXT 10

Col6=REGION_LBL,TEXT 50


```
; Match fields in flat file with levels and labels in dimension
```

```
Level1=CUSTOMER
```

```
Level2=DISTRICT
```

```
Level3=REGION
```

```
Label1=CUSTOMER_LBL
```

```
Label2=DISTRICT_LBL
```

```
Label3=REGION_LBL
```

```
; Attribute load with source data in relational
```

```
; Have to build dimension component first
```

```
[cot_dim_sql]
```

```
Loading=Dimension
```

```
Schema=lss
```

```
Table=cot
```

```
LinkIDSource=orasource
```

```
SourceSQL=select COT from customer
```

```
Format=Delimited TAB
```

```
LinkIDTarget=holap
```

```
Result=Yes
```

```
Truncate=Yes
```

```
Trace=No
```

```
Level1=COT
```

```
Label1=COT
```

```
; Load the data for the attribute variable
```

```
[cot_data_sql]
```

```
Loading=Attribute
```

```
Schema=lss
```

```
Table=cot
```

```
Dimension=customer
```

```
LinkIDSource=orasource
```

```
SourceSQL=select CUSTOMER, COT from customer
```

```
Format=Delimited TAB
```

```
LinkIDTarget=holap
```

```
Truncate=Yes
```

```
Trace=No
```

DimField=CUSTOMER

AttrField=COT

; Initial data load from SQL source

[data_sql_init]

Loading=Fact

Schema=lss

LinkIDSource=orasource

SourceSQL=select CUSTOMER, PRODUCT, CHANNEL,
to_char(NEWDATE,'yyyy/mm/dd') NEWTIME, UNITS, SALES, COSTS from
transact_hist

Format=Delimited TAB

DateFormat=YYYY/MM/DD

LinkIDTarget=holap

Truncate=Yes

Trace=No

Blocks=250000

BlockSize=8192

Load=10000

; Match returned fields with model components

VARIABLE1=UNITS,UNITS

VARIABLE2=SALES,SALES

VARIABLE3=COSTS,COSTS

DIM1=CUSTOMER,CUSTOMER

DIM2=CHANNEL,CHANNEL

DIM3=PRODUCT,PRODUCT

TIME=NEWTIME

; Incremental data load from SQL source

[data_sql_incr]

Loading=Fact

Schema=lss

LinkIDSource=orasource

SourceSQL=select CUSTOMER, PRODUCT, CHANNEL,
to_char(NEWDATE,'yyyy/mm/dd') NEWTIME, UNITS, SALES, COSTS from
transact_currmonth

Format=Delimited TAB

```
DateFormat=YYYY/MM/DD
LinkIDTarget=holap
Truncate=No
Period=2003/01/01-2003/01/01
Trace=No
Blocks=250000
BlockSize=8192
Load=10000

; Match returned fields with components in Application Server model
VARIABLE1=UNITS,UNITS
VARIABLE2=SALES,SALES
VARIABLE3=COSTS,COSTS
DIM1=CUSTOMER,CUSTOMER
DIM2=CHANNEL,CHANNEL
DIM3=PRODUCT,PRODUCT
TIME=NEWTIME

; Initial data load from data in flat file
[data_text_init]
Loading=Fact
Schema=lss
InputFile=transact.txt
Format=Delimited Tab
DateFormat=MM/DD/YYYY
LinkIDTarget=holap
Truncate=Yes
Trace=No
Blocks=250000
BlockSize=8192
Load=10000

; Describe columns in flat file
COL1=CUSTOMER,TEXT 24
COL2=PRODUCT,TEXT 24
COL3=CHANNEL,TEXT 24
COL4=TIME,TEXT 10
COL5=UNITS,NUMERIC 10.0
```

Parameter File Reference

COL6=SALES,NUMERIC 10.2

COL7=COSTS,NUMERIC 10.2

VARIABLE1=UNITS,UNITS

VARIABLE2=SALES,SALES

VARIABLE3=COSTS,COSTS

DIM1=CUSTOMER,CUSTOMER

DIM2=CHANNEL,CHANNEL

DIM3=PRODUCT,PRODUCT

TIME=TIME

11.5.2 Parameters: All Sections

This topic contains the parameters that are valid for all sections of the Transformer parameter file.

The first table describes parameters that are required.

The second table shows optional parameters that have default values if not used.

The third table shows optional parameters that can be used in applicable situations. They do not have default values if they are not used.

(Table 1) Keyword/Syntax **Description of Required parameters**

[windows] TBDB= <i>tddb_path</i>	where <i>tddb_path</i> is the full path and name of the TBDB.ENG file which stores Transformer messages string. This section is required on Windows NT systems. On UNIX define TBDB as an environment variable prior to running the Transformer.
[<i>section_name</i>]	where <i>section_name</i> is a descriptive label for the section of the file. When you run the Transformer you pass the name of the parameter file and the name of the section as follows. <code>sgtrans -inifile <i>file_name</i> -s <i>section_name</i></code>
Loading= <i>table_type</i>	where <i>table_type</i> is the type of table that is being loaded. The setting is Dimension, Fact, or Attribute.
Schema= <i>prefix</i>	where <i>prefix</i> is the schema prefix of the table(s) to be updated.
LinkIDSource= <i>LinkID</i> and	where <i>LinkID</i> is a valid Link ID to connect to source data residing in a supported RDBMS.
SourceSQL=SQL	where <i>SQL</i> is the SQL statement(s) to retrieve the required data from the source data in the RDBMS.
or	
InputFile= <i>file_name</i>	where <i>file_name</i> is the full path and name of the input ASCII text file. Note: A Transformer parameter file includes either LinkIDSource and SourceSQL or InputFile, but not both
Format= <i>style</i>	where <i>style</i> defines the field layout of the source file. The setting is one of the following: Fixed, Delimited Tab, or Delimited, (to indicate a comma-delimited format). Other field separator characters are accepted, for example, Delimited~ and Delimited .
LinkIDTarget= <i>LinkID</i>	where <i>LinkID</i> is a valid Link ID to connect to the schema tables in the RDBMS.
Col1= <i>colname,coltype length</i> to	where <i>colname</i> is the alias for the field name, <i>coltype</i> is text for dimension fields and numeric for variable fields. For text fields <i>length</i> is the number of characters and for numeric fields it is the number of digits to the left and right of the decimal (ex. 10.2)
Coln= <i>colname,coltype length</i>	

These entries are required when the source data is from an ASCII text file. There must be one *Coln=* entry for each column in the file.

Note: You must insert a comma between *colname* and *coltype* and a space between *coltype* and *length*.

This table below contains the syntax for optional parameters in all sections. If you do not specify these parameters, default values will be used.

(Table 2) Keyword/Syntax	Description of optional parameters with default values
Truncate=yes no	where <i>no</i> indicates that updates will be added to the existing table and <i>yes</i> truncates or drops existing tables prior to the load. Use <i>No</i> for an incremental update and <i>Yes</i> when rebuilding the schema. If you do not specify this parameter, then Truncate=no is used.
Trace=yes no	where <i>yes</i> displays the SQL being executed to the terminal. If you do not specify this parameter, then Trace=no is used.
Blocksize= <i>n</i>	where <i>n</i> is the blocksize of the Transformer cache. If you do not specify this parameter, then Blocksize=2048 is used.
Blocks= <i>n</i>	where <i>n</i> is the number of blocks in the Transformer cache. Blocks is used in conjunction with Blocksize to determine the size to which the cache file can grow. If you do not specify this parameter, then Blocks=25000 is used.
Load= <i>n</i>	where <i>n</i> is the number of SQL target records to buffer for output to the SQL fast loader. If you do not specify this parameter, then Load=1000 is used.
WorkDB= <i>file_name</i>	where <i>file_name</i> is the name of the temporary cache file created by the Transformer. If you specify a unique cache file name in each section, you can run multiple Transformer sessions simultaneously. If you do not specify this parameter, then WorkDB=SGTRANSC is used.
Timings=Yes No	Controls whether to display or remove the timings that are shown in Transformer output when the Trace parameter is set to Yes. To display timings, specify Timings=Yes. To omit timings, specify Timings=No. If you do not specify this parameter and Trace=Yes, then Timings=No is used and no timings are displayed during a trace.
Nthreads= <i>n</i>	Specifies the number of threads to use. If Nthreads=1, the Transformer and the RDBMS share one thread. If you specify Nthreads=2, two threads are started and the tasks can operate concurrently. If you do not specify this parameter, then Nthreads=1 is used and the Transformer and RDBMS share one thread.
OutputFile= <i>file_name</i>	where <i>file_name</i> is the full path and name of the output ASCII text file. If an output file is defined, transformed data is output to an ASCII text file and is not loaded into the schema. If no output file is defined or if you do not specify this parameter, then data is loaded into the schema tables.

This table below contains the syntax for optional parameters in all sections. Use these optional parameters only when applicable to the situation. If you do not specify these parameters, no values will be used.

(Table 3) Keyword/Syntax	Description of optional parameters with no default values
TryFirst= INSERT UPDATE	Specifies whether Application Server will try to insert rows first or update rows first. If you know that your data will be adding new records to the relational tables, specify TryFirst=Insert to improve

performance. If you know that your data will be updating records that are in the tables already, specify TryFirst=Update.

This keyword is used only for Fact tables.

By default Transformer will try inserts first when the Schema is storing time down and it will try updates first when the Schema is storing time across.

When the TRUNCATE=YES parameter is used, then TryFirst=INSERT will always be used.

Comment=*character* where *character* is the character used in the input ASCII text file to indicate a comment line that should be skipped during the load.

Delimited=*character* where *character* specifies the quote character that surrounds input data strings in ASCII input files. If the strings contain the field delimiter character then they must be in quotes.

Directload=Yes|No Indicates whether or not to use Oracle Sql*Loader to perform a direct path load of the Dimension or Fact table.

Direct path load is valid only when creating the tables. To use Directload=Yes, the parameter Truncate=Yes must be used.

11.5.3 Parameters: Dimension Sections

This table contains the syntax for the required parameters in Dimension sections.

The first table describes parameters that are required.

The second table shows optional parameters that have default values if not used.

(Table 1) Keyword/Syntax Description of required parameters

Table=dimension	where dimension is the name of the structural or attribute dimension to be loaded or updated.
Result=yes/no	where yes adds a Result (apex) member and level to the dimension.
Level1=field	where field names the column from the source data that represents the short names for the level in the dimension. Level1 corresponds to the input members of the dimension and Leveln to the highest output level below the Result member.
To	
[Leveln=field]	
Label1=field	where field names the column from the source data that represents the long names or labels for the members at the corresponding level of the dimension.
To	
[Labeln=field]	
Note: For Level and Label entries some manipulation of the text values is possible. Instead of naming a column in the source data, field may be a literal string enclosed in quotation marks (") or a substring or a column. It also may be a concatenation of two or more of these elements separated by plus signs (+). The syntax for the substring function in the Parameter reference file is: SUBSTR(column_name, offset, length)	

This table below contains the syntax for optional parameters in the Dimension sections. If you do not specify these parameters, default values will be used.

(Table 2) Keyword/Syntax Description of optional parameters with default

Delete=Yes No	For Dimension table loads, this parameter indicates that the members in the source should be deleted from the schema tables. If you do not specify this parameter, then Delete=No will be used.
Cachedim=Yes No	For Dimension table updates, setting this parameter to Yes causes the

Transformer to load a copy of existing members into memory prior to the update.

This may speed up the load in cases where there are a lot of records in the new data set. If the update data set is small, setting this parameter to No may improve the load time.

If you do not specify this parameter, then Cachedim=No will be used.

11.5.4 Parameters: Fact Sections

This table contains the syntax for the required parameters in Fact sections.

The first table describes parameters that are required.

The second table shows optional parameters that have default values if not used.

(Table 1) Keyword/Syntax Description of required parameters

DateFormat= <i>date_format</i>	where <i>date_format</i> is the format of the dates in the source data.
Period=start_date [-end_date]	where <i>start_date</i> is the date of the first observation to load, and <i>end_date</i> is the optional date of the last observation to load. <i>start_date</i> and <i>end_date</i> must be entered in the format specified in the DateFormat setting. The default for <i>end_date</i> is <i>start_date</i> . Note: The Period setting should be used when you are updating an already-populated Fact table and you have Truncate set to No.
Variable1= <i>var_name, field</i> To Variablen= <i>var_name, field</i>	where <i>var_name</i> is the Application Server short name for the variable being loaded and <i>field</i> is the corresponding column in the source data. There is one Variablen= entry for every variable in the source data. Note: All variables loaded together must have the same dimensionality and periodicity.
Dim1= <i>dim_name, field</i> To Dimn= <i>dim_name, field</i>	where <i>dim_name</i> specifies the dimensions being loaded and <i>field</i> is the corresponding column in the source data. All of the dimensions that the variables are dimensioned by must be listed. Note: The string manipulation described above for Level and Label entries is an option here also.
Time= <i>field</i>	where <i>field</i> maps to the source data column that contains the date.

This table below contains the syntax for optional parameters in the Fact sections. If you do not specify these parameters, default values will be used.

(Table 2) Keyword/Syntax Description of optional parameter with default value

ReadOption=Add	During Fact table loads, ReadOption=Add aggregates input data if required, and adds it to corresponding observations that already exist in the Fact table. New rows are created in the Fact table if required. If this parameter is used, you must set the Truncate parameter to Truncate=No. If you do not specify this parameter, data will not be aggregated.
----------------	--

11.5.5 Parameters: Attribute Sections

The attribute dimension is built first in the same manner as a structural dimension and then data is loaded for the attribute variable. The following table contains the syntax for additional entries required for Attribute variable sections:

Keyword/Syntax	Description
Table=attribute	where attribute is the name of the attribute dimension to be loaded or

	updated.
Dimension=dimension	where dimension is the structural dimension this attribute is tied to.
DimField=field	where field is the column in the source data that represents the short name for the input values of the structural dimension.
AttrField=field	where field is the column in the source data that represents the attribute data value associated with the structural dimension member in DimField.

11.5.6 Tips on using the Transformer

Keep the following points in mind as you use the Transformer:

Do not add unnecessary spaces to parameter file settings. If you are not sure, follow the examples in this Help file.

Give each output ASCII text file you create a unique name. OutputFile does not overwrite an existing file with the same name.

To get the best performance out of initial data loads via the Transformer, disable triggers on your schema tables before the Transformer run, and re-enable them afterwards, using the TRIGGER (Schema) command. That way, you avoid needlessly populating the Fact Delta table(s).